UM1755
User manual

BlueNRG Bluetooth® LE stack application command interface (ACI)

Introduction

The purpose of this document is to describe the design of the application command interface (ACI) of the BlueNRG Bluetooth low energy (LE) stack. This document specifies the list of commands supported by the BlueNRG ACI.

Contents

1 Overview 14

1.1 BlueNRG ACI architecture 15

2 ACI command data format 16

2.1 Overview 16

2.2 HCI command packet 16

2.3 HCI event packet 18

2.4 HCI ACL data packet 18

3 Standard HCI commands 19

3.1 Supported HCI commands 19

3.2 Supported HCI events 20

4 Vendor specific commands 21

4.1 VS command and VS event format 21

4.2 L2CAP VS commands and events 23

4.2.1 L2CAP VS commands 23

4.2.2 Aci_L2CAP_Connection_Parameter_Update_Request 23

4.2.3 Aci_L2CAP_Connection_Parameter_Update_Response 24

4.2.4 L2CAP VS events 25

4.2.5 Evt_Blue_L2CAP_Connection_Update_Resp 25

4.2.6 Evt_Blue_L2CAP_Procedure_Timeout 26

4.2.7 Evt_Blue_L2CAP_Connection_Update_Request 26

4.3 GAP VS commands and events 27

4.3.1 Overview 27

4.3.2 GAP VS Commands 28

4.3.3 Aci_Gap_Set_Non_Discoverable 29

4.3.4 Aci_Gap_Set_Limited_Discoverable 29

4.3.5 Aci_Gap_Set_Discoverable 31

4.3.6 Aci_Gap_Set_Direct_Connectable 33

4.3.7 Aci_Gap_Set_IO_Capability 34

4.3.8 Aci_Gap_Set_Auth_Requirement 35

4.3.9 Aci_Gap_Set_Author_Requirement 36

4.3.10 Aci_Gap_Pass_Key_Response . . . 37
4.3.11 Aci_Gap_Authorization_Response . . . 37
4.3.12 Aci_Gap_Init . . . 38
4.3.13 Aci_Gap_Set_Non_Connectable . . . 39
4.3.14 Aci_Gap_Set_Undirected_Connectable . . . 40
4.3.15 Aci_Gap_Slave_Security_Request . . . 41
4.3.16 Aci_Gap_Update_Adv_Data . . . 42
4.3.17 Aci_Gap_Delete_AD_Type . . . 42
4.3.18 Aci_Gap_Get_Security_Level . . . 43
4.3.19 Aci_Gap_Set_Event_Mask . . . 45
4.3.20 Aci_Gap_Configure_WhiteList . . . 46
4.3.21 Aci_Gap_Terminate . . . 46
4.3.22 Aci_Gap_Clear_Security_Database . . . 47
4.3.23 Aci_Allow_Rebond . . . 48
4.3.24 Aci_Gap_Start_Limited_Discovery_Proc . . . 48
4.3.25 Aci_Gap_Start_General_Discovery_Proc . . . 50
4.3.26 Aci_Gap_Start_Name_Discovery_Proc . . . 51
4.3.27 Aci_Gap_Start_Auto_Conn_Establishment . . . 53
4.3.28 Aci_Gap_Start_General_Conn_Establishment . . . 55
4.3.29 Aci_Gap_Start_Selective_Conn_Establishment . . . 56
4.3.30 Aci_Gap_Create_Connection . . . 58
4.3.31 Aci_Gap_Terminate_Gap_Procedure . . . 59
4.3.32 Aci_Gap_Start_Connection_Update . . . 61
4.3.33 Aci_Gap_Send_Pairing_Request . . . 62
4.3.34 Aci_Gap_Resolve_Private_Address . . . 63
4.3.35 Aci_Gap_Get_Bonded_Devices . . . 64
4.4 GAP VS events . . . 64
4.4.1 Evt_Blue_Gap_Limited_Discoverable . . . 65
4.4.2 Evt_Blue_Gap_Pairing_Complete . . . 65
4.4.3 Evt_Blue_Pass_Key_Request . . . 65
4.4.4 Evt_Blue_Authorization_Request . . . 67
4.4.5 Evt_Blue_Slave_Security_Initiated . . . 67
4.4.6 Evt_Blue_Gap_Bond_Lost . . . 67
4.4.7 Evt_Blue_Gap_Device_Found . . . 67
4.4.8 Evt_Blue_Gap_Procedure_Complete . . . 68
4.4.9 Evt_Blue_Gap_Reconnection_Address . . . 69
4.5 GATT VS commands and events . . . 69

4.5.1 GATT VS commands . . . 69
4.5.2 Aci_Gatt_Init . . . 71
4.5.3 Aci_Gatt_Add_Service . . . 71
4.5.4 Aci_Gatt_Include_Service . . . 72
4.5.5 Aci_Gatt_Add_Characteristic . . . 73
4.5.6 Aci_Gatt_Add_Char_Desc . . . 75
4.5.7 Aci_Gatt_Update_Char_Value . . . 77
4.5.8 Aci_Gatt_Del_Characteristic . . . 78
4.5.9 Aci_Gatt_Del_Service . . . 79
4.5.10 Aci_Gatt_Del_Include_Service . . . 80
4.5.11 Aci_Gatt_Set_Event_Mask . . . 80
4.5.12 Aci_Gatt_Exchange_Configuration . . . 81
4.5.13 Aci_Att_Find_Information_Req . . . 82
4.5.14 Att_Find_By_Type_Value_Req . . . 83
4.5.15 Aci_Att_Read_Attr_By_Type_Req . . . 84
4.5.16 Aci_Att_Read_By_Group_Type_Req . . . 85
4.5.17 Aci_Att_Prepare_Write_Req . . . 86
4.5.18 Aci_Att_Execute_Write_Req . . . 87
4.5.19 Aci_Gatt_Disc_All_Prim_Services . . . 88
4.5.20 Aci_Gatt_Disc_Prim_serv_By_Uuid . . . 89
4.5.21 Aci_Gatt_Find_Included_Services . . . 90
4.5.22 Aci_Gatt_Disc_All_Charac_Of_Serv . . . 91
4.5.23 Aci_Gatt_Disc_Charac_By_UUID . . . 92
4.5.24 Aci_Gatt_Disc_All_Charac_Descriptors . . . 93
4.5.25 Aci_Gatt_Read_Charac_Val . . . 94
4.5.26 Aci_Gatt_Read_Charac_Using_UUID . . . 95
4.5.27 Aci_Gatt_Read_Long_Charac_Val . . . 96
4.5.28 Aci_Gatt_Read_Multiple_Charac_Values . . . 97
4.5.29 Aci_Gatt_Write_Charac_Val . . . 98
4.5.30 Aci_Gatt_Write_Long_Charac_Val . . . 99
4.5.31 Aci_Gatt_Write_Charac_Reliable . . . 100
4.5.32 Aci_Gatt_Write_Long_Charac_Descriptor . . . 101
4.5.33 Aci_Gatt_Read_Long_Charac_Descriptor . . . 102
4.5.34 Aci_Gatt_Write_Charac_Descriptor . . . 103
4.5.35 Aci_Gatt_Read_Charac_Descriptor . . . 104
4.5.36 Aci_Gatt_Write_Without_Response . . . 104
4.5.37 Aci_Gatt_Signed_Write_Without_Response . . . 105

4.5.38 Aci_Gatt_Confirm_Indication . . . 106
4.5.39 Aci_Gatt_Write_Resp . . . 107
4.5.40 Aci_Gatt_Allow_Read . . . 108
4.5.41 Aci_Gatt_Set_Security_Permission . . . 109
4.5.42 Aci_Gatt_Set_Descriptor_Value . . . 110
4.5.43 Aci_Gatt_Read_Handle_Value . . . 111
4.6 GATT VS events . . . 112
4.6.1 Evt_Blue_Gatt_Attribute_modified . . . 112
4.6.2 Evt_Blue_Gatt_Procedure_Timeout . . . 113
4.6.3 Evt_Blue_Gatt_Procedure_Complete . . . 113
4.6.4 Evt_Blue_Gatt_Disc_Read_Charac_By_UUID_Resp . . . 113
4.6.5 Evt_Blue_Gatt_Write_Permit_req . . . 114
4.6.6 Evt_Blue_Gatt_Read_Permit_Req . . . 114
4.6.7 Evt_Blue_Gatt_Read_Multi_Permit_Req . . . 115
4.6.8 Other GATT VS events . . . 115
4.7 HCI Vendor specific commands . . . 116
4.7.1 HCI VS commands . . . 116
4.7.2 Hal_IO_Init . . . 117
4.7.3 Hal_IO_Configure . . . 117
4.7.4 Hal_IO_set_bit . . . 118
4.7.5 Hal_IO_get_bit . . . 118
4.7.6 Write_Config_Data . . . 119
4.7.7 Read_Config_Data . . . 120
4.7.8 Set_Tx_power_level . . . 121
4.7.9 Device_Standby . . . 122
4.7.10 Number_Of_Packet_Sent_in_DirectTX . . . 123
4.7.11 Tone_Start . . . 123
4.7.12 Tone_Stop . . . 124

5 SPI interface . . . 125
5.1 Hardware SPI interface . . . 125
5.2 SPI communication protocol . . . 127
5.2.1 Header . . . 127
5.2.2 Write data to BlueNRG . . . 128
5.2.3 Read data from BlueNRG . . . 128
5.2.4 SPI operation with BlueNRG sleep mode . . . 129

6 Revision history 130

List of tables

Table 1. OGF value . . . 17
Table 2. HCI commands . . . 19
Table 3. HCI events . . . 20
Table 4. CGID group . . . 21
Table 5. EGID group . . . 22
Table 6. L2CAP VS commands . . . 23
Table 7. Aci_L2CAP_Connection_Parameter_Update_Request . . . 23
Table 8. Aci_L2CAP_Connection_Parameter_Update_Requests command parameters . . . 23
Table 9. Aci_L2CAP_Connection_Parameter_Update_Requests return parameters . . . 24
Table 10. Aci_L2CAP_Connection_Parameter_Update_Response . . . 24
Table 11. Aci_L2CAP_Connection_Parameter_Update_Response command parameters . . . 24
Table 12. Aci_L2CAP_Connection_Parameter_Update_Response return parameters . . . 25
Table 13. L2CAP VS events . . . 25
Table 14. Evt_Blue_L2CAP_Connection_Update_Resp . . . 25
Table 15. Evt_Blue_L2CAP_Procedure_Timeout . . . 26
Table 16. Evt_Blue_L2CAP_Connection_Update_Request . . . 26
Table 17. GAP VS commands . . . 28
Table 18. Aci_gap_set_non_discoverable . . . 29
Table 19. Aci_gap_set_non_discoverable return parameters . . . 29
Table 20. Aci_gap_set_limited_discoverable . . . 29
Table 21. Aci_gap_set_limited_discoverable command parameters . . . 30
Table 22. Aci_gap_set_limited_discoverable return parameters . . . 31
Table 23. Aci_gap_set_discoverable . . . 31
Table 24. Aci_gap_set_discoverable command parameters . . . 31
Table 25. Aci_gap_set_discoverable return parameters . . . 32
Table 26. Aci_Gap_Set_Direct_Connectable . . . 33
Table 27. Aci_Gap_Set_Direct_Connectable command parameters . . . 33
Table 28. Aci_Gap_Set_Direct_Connectable return parameters . . . 33
Table 29. Aci_Gap_Set_IO_Capability . . . 34
Table 30. Aci_Gap_Set_IO_Capability command parameters . . . 34
Table 31. Aci_Gap_Set_IO_Capability return parameters . . . 34
Table 32. Aci_Gap_Set_Auth_Requirement . . . 35
Table 33. Aci_Gap_Set_Auth_Requirement command parameters . . . 35
Table 34. Aci_Gap_Set_Auth_Requirement return parameters . . . 36
Table 35. Aci_Gap_Set_Author_Requirement . . . 36
Table 36. Aci_Gap_Set_Author_Requirement command parameters . . . 36
Table 37. Aci_Gap_Set_Author_Requirement return parameters . . . 36
Table 38. Aci_Gap_Pass_Key_Response . . . 37
Table 39. Aci_Gap_Pass_Key_Response command parameters . . . 37
Table 40. Aci_Gap_Pass_Key_Response return parameters . . . 37
Table 41. Aci_Gap_Authorization_Response . . . 37
Table 42. Aci_Gap_Authorization_Response command parameters . . . 38
Table 43. Aci_Gap_Authorization_Response return parameters . . . 38
Table 44. Aci_Gap_Init . . . 38
Table 45. Aci_Gap_Init command parameters . . . 38
Table 46. Aci_Gap_Init return parameters . . . 39
Table 47. Aci_Gap_Set_Non_Connectable . . . 39
Table 48. Aci_Gap_Set_Non_Connectable command parameters . . . 39

Table 49. Aci_Gap_Set_Non_Connectable return parameters . . . 39
Table 50. Aci_Gap_Set_Undirected_Connectable . . . 40
Table 51. Aci_Gap_Set_Undirected_Connectable command parameters . . . 40
Table 52. Aci_Gap_Set_Undirected_Connectable return parameters . . . 40
Table 53. Aci_Gap_Slave_Security_Request . . . 41
Table 54. Aci_Gap_Slave_Security_Request command parameters . . . 41
Table 55. Aci_Gap_Slave_Security_Request return parameters . . . 41
Table 56. Aci_Gap_Update_Adv_Data . . . 42
Table 57. Aci_Gap_Update_Adv_Data command parameters . . . 42
Table 58. Aci_Gap_Update_Adv_Data return parameters . . . 42
Table 59. Aci_Gap_Delete_AD_Type . . . 42
Table 60. Aci_Gap_Delete_AD_Type command parameters . . . 43
Table 61. Aci_Gap_Delete_AD_Type return parameters . . . 43
Table 62. Aci_Gap_Get_Security_Level . . . 43
Table 63. Aci_Gap_Get_Security_Level return parameters . . . 44
Table 64. Aci_Gap_Set_Event_Mask . . . 45
Table 65. Aci_Gap_Set_Event_Mask command parameters . . . 45
Table 66. Aci_Gap_Set_Event_Mask return parameters . . . 45
Table 67. Aci_Gap_Configure_WhiteList . . . 46
Table 68. Aci_Gap_Configure_WhiteList return parameters . . . 46
Table 69. Aci_Gap_Terminate . . . 46
Table 70. Aci_Gap_Terminate command parameters . . . 46
Table 71. Aci_Gap_Terminate return parameters . . . 47
Table 72. Aci_Gap_Clear_Security_Database . . . 47
Table 73. Aci_Gap_Clear_Security_Database return parameters . . . 47
Table 74. Aci_Allow_Rebond . . . 48
Table 75. Aci_Allow_Rebond return parameters . . . 48
Table 76. Aci_Gap_Start_Limited_Discovery_Proc . . . 48
Table 77. Aci_Gap_Start_Limited_Discovery_Proc command parameters . . . 48
Table 78. Aci_Gap_Start_Limited_Discovery_Proc return parameters . . . 49
Table 79. Aci_Gap_Start_General_Discovery_Proc . . . 50
Table 80. Aci_Gap_Start_General_Discovery_Proc command parameters . . . 50
Table 81. Aci_Gap_Start_General_Discovery_Proc return parameters . . . 50
Table 82. Aci_Gap_Start_Name_Discovery_Proc . . . 51
Table 83. Aci_Gap_Start_Name_Discovery_Proc command parameters . . . 51
Table 84. Aci_Gap_Start_Name_Discovery_Proc return parameters . . . 52
Table 85. Aci_Gap_Start_Auto_Conn_Establishment . . . 53
Table 86. Aci_Gap_Start_Auto_Conn_Establishment command parameters . . . 53
Table 87. Aci_Gap_Start_Auto_Conn_Establishment return parameters . . . 54
Table 88. Aci_Gap_Start_General_Conn_Establishment . . . 55
Table 89. Aci_Gap_Start_General_Conn_Establishment command parameters . . . 55
Table 90. Aci_Gap_Start_General_Conn_Establishment return parameters . . . 56
Table 91. Aci_Gap_Start_Selective_Conn_Establishment . . . 56
Table 92. Aci_Gap_Start_Selective_Conn_Establishment command parameters . . . 56
Table 93. Aci_Gap_Start_General_Conn_Establishment return parameters . . . 57
Table 94. Aci_Gap_Create_Connection . . . 58
Table 95. Aci_Gap_Create_Connection command parameters . . . 58
Table 96. Aci_Gap_Create_Connection return parameters . . . 59
Table 97. Aci_Gap_Terminate_Gap_Procedure . . . 59
Table 98. Aci_Gap_Terminate_Gap_Procedure command parameters . . . 60
Table 99. Aci_Gap_Terminate_Gap_Procedure return parameters . . . 60
Table 100. Aci_Gap_Start_Connection_Update . . . 61

Table 101. Aci_Gap_Start_Connection_Update command parameters . . . 61
Table 102. Aci_Gap_Start_Connection_Update return parameters . . . 62
Table 103. Aci_Gap_Send_Pairing_Request . . . 62
Table 104. Aci_Gap_Send_Pairing_Request command parameters . . . 62
Table 105. Aci_Gap_Send_Pairing_Request return parameters . . . 63
Table 106. Aci_Gap_Resolve_Private_Address . . . 63
Table 107. Aci_Gap_Resolve_Private_Address command parameters . . . 63
Table 108. Aci_Gap_Resolve_Private_Address return parameters . . . 63
Table 109. Aci_Gap_Get_Bonded_Devices . . . 64
Table 110. Aci_Gap_Get_Bonded_Devices return parameters . . . 64
Table 111. GAP VS events . . . 64
Table 112. Evt_Blue_Gap_Pairing_Complete . . . 65
Table 113. Evt_Blue_Pass_Key_Request . . . 66
Table 114. Evt_Blue_Authorization_Request . . . 67
Table 115. Evt_Blue_Gap_Device_Found . . . 67
Table 116. Evt_Blue_Gap_Procedure_Complete . . . 68
Table 117. Evt_Blue_Gap_Reconnection_Address . . . 69
Table 118. GATT VS commands . . . 69
Table 119. Aci_Gatt_Init . . . 71
Table 120. Aci_Gatt_Init return parameters . . . 71
Table 121. Aci_Gatt_Add_Service . . . 71
Table 122. Aci_Gatt_Add_Service command parameters . . . 71
Table 123. Aci_Gatt_Add_Service return parameters . . . 72
Table 124. Aci_Gatt_Include_Service . . . 72
Table 125. Aci_Gatt_Include_Service command parameters . . . 73
Table 126. Aci_Gatt_Include_Service return parameters . . . 73
Table 127. Aci_Gatt_Add_Characteristic . . . 73
Table 128. Aci_Gatt_Add_Characteristic command parameters . . . 74
Table 129. Aci_Gatt_Add_Characteristic return parameters . . . 75
Table 130. Aci_Gatt_Add_Char_Desc . . . 75
Table 131. Aci_Gatt_Add_Char_Desc command parameters . . . 76
Table 132. Aci_Gatt_Add_Char_Desc return parameters . . . 77
Table 133. Aci_Gatt_Update_Char_Value . . . 77
Table 134. Aci_Gatt_Update_Char_Value command parameters . . . 77
Table 135. Aci_Gatt_Update_Char_Value return parameters . . . 78
Table 136. Aci_Gatt_Del_Characteristic . . . 78
Table 137. Aci_Gatt_Del_Characteristic command parameters . . . 78
Table 138. Aci_Gatt_Del_Characteristic return parameters . . . 79
Table 139. Aci_Gatt_Del_Service . . . 79
Table 140. Aci_Gatt_Del_Service command parameters . . . 79
Table 141. Aci_Gatt_Del_Service return parameters . . . 79
Table 142. Aci_Gatt_Del_Include_Service . . . 80
Table 143. Aci_Gatt_Del_Include_Service command parameters . . . 80
Table 144. Aci_Gatt_Del_Include_Service return parameters . . . 80
Table 145. Aci_Gatt_Set_Event_Mask . . . 80
Table 146. Aci_Gatt_Set_Event_Mask command parameters . . . 81
Table 147. Aci_Gatt_Set_Event_Mask return parameters . . . 81
Table 148. Aci_Gatt_Exchange_Configuration . . . 81
Table 149. Aci_Gatt_Exchange_Configuration command parameters . . . 82
Table 150. Aci_Gatt_Exchange_Configuration return parameters . . . 82
Table 151. Aci_Att_Find_Information_Req . . . 82
Table 152. Aci_Att_Find_Information_Req command parameters . . . 82

Table 153. Aci_Att_Find_Information_Req return parameters . . . 83
Table 154. Att_Find_By_Type_Value_Req . . . 83
Table 155. Aci_Att_Find_Attr_By_Typ_And_Value_Req command parameters . . . 83
Table 156. Aci_Att_Find_Attr_By_Typ_And_Value_Req return parameters . . . 84
Table 157. Aci_Att_Read_Attr_By_Type_Req . . . 84
Table 158. Aci_Att_Read_Attr_By_Type_Req command parameters . . . 85
Table 159. Aci_Att_Find_Attr_By_Typ_And_Value_Req return parameters . . . 85
Table 160. Aci_Att_Read_By_Group_Type_Req . . . 85
Table 161. Aci_Att_Read_Attr_By_Grp_Type_Req command parameters . . . 86
Table 162. Aci_Att_Read_Attr_By_Grp_Type_Req return parameters . . . 86
Table 163. Aci_Att_Prepare_Write_Req . . . 86
Table 164. Aci_Att_Prepare_Write_Req command parameters . . . 87
Table 165. Aci_Att_Prepare_Write_Req return parameters . . . 87
Table 166. Aci_Gatt_Execute_Write_Req . . . 87
Table 167. Aci_Att_Execute_Write_Req command parameters . . . 88
Table 168. Aci_Att_Execute_Write_Req return parameters . . . 88
Table 169. Aci_Gatt_Disc_All_Prim_Services . . . 88
Table 170. Aci_Gatt_Disc_All_Prim_Services command parameters . . . 88
Table 171. Aci_Gatt_Disc_All_Prim_Services return parameters . . . 89
Table 172. Aci_Gatt_Disc_Prim_serv_By_Uuid . . . 89
Table 173. Aci_Gatt_Disc_Prim_serv_By_Uuid command parameters . . . 89
Table 174. Aci_Gatt_Disc_Prim_serv_By_Uuid return parameters . . . 90
Table 175. Aci_Gatt_Find_Included_Services . . . 90
Table 176. Aci_Gatt_Find_Included_Services command parameters . . . 90
Table 177. Aci_Gatt_Find_Included_Services return parameters . . . 91
Table 178. Aci_Gatt_Disc_All_Charac_Of_Serv . . . 91
Table 179. Aci_Gatt_Disc_All_Charac_Of_Serv command parameters . . . 91
Table 180. Aci_Gatt_Disc_All_Charac_Of_Serv return parameters . . . 92
Table 181. Aci_Gatt_Disc_Charac_By_UUID . . . 92
Table 182. Aci_Gatt_Disc_Charac_By_UUID command parameters . . . 92
Table 183. Aci_Gatt_Disc_Charac_By_UUID return parameters . . . 93
Table 184. Aci_Gatt_Disc_All_Charac_Descriptors . . . 93
Table 185. Aci_Gatt_Disc_All_Charac_Descriptors command parameters . . . 93
Table 186. Aci_Gatt_Disc_All_Charac_Descriptors return parameters . . . 94
Table 187. Aci_Gatt_Read_Charac_Val . . . 94
Table 188. Aci_Gatt_Read_Charac_Val command parameters . . . 94
Table 189. Aci_Gatt_Read_Charac_Val return parameters . . . 94
Table 190. Aci_Gatt_Read_Charac_Using_UUID . . . 95
Table 191. Aci_Gatt_Read_Charac_Using_UUID command parameters . . . 95
Table 192. Aci_Gatt_Read_Charac_Using_UUID return parameters . . . 95
Table 193. Aci_Gatt_Read_Long_Charac_Val . . . 96
Table 194. Aci_Gatt_Read_Long_Charac_Val command parameters . . . 96
Table 195. Aci_Gatt_Read_Long_Charac_Val command return parameters . . . 96
Table 196. Aci_Gatt_Read_Multiple_Charac_Values . . . 97
Table 197. Aci_Gatt_Read_Multiple_Charac_Values command parameters . . . 97
Table 198. Aci_Gatt_Read_Multiple_Charac_Values return parameters . . . 97
Table 199. Aci_Gatt_Write_Charac_Val . . . 98
Table 200. Aci_Gatt_Write_Charac_Val command parameters . . . 98
Table 201. Aci_Gatt_Write_Charac_Val return parameters . . . 98
Table 202. Aci_Gatt_Write_Long_Charac_Val . . . 99
Table 203. Aci_Gatt_Write_Long_Charac_Val command parameters . . . 99
Table 204. Aci_Gatt_Write_Long_Charac_Val return parameters . . . 99

Table 205. Aci_Gatt_Write_Charac_Reliable . . . 100
Table 206. Aci_Gatt_Write_Charac_Reliable command parameters . . . 100
Table 207. Aci_Gatt_Write_Charac_Reliable return parameters . . . 100
Table 208. Aci_Gatt_Write_Long_Charac_Descriptor . . . 101
Table 209. Aci_Gatt_Write_Long_Charac_Descriptor command parameters . . . 101
Table 210. Aci_Gatt_Write_Long_Charac_Descriptor return parameters . . . 101
Table 211. Aci_Gatt_Read_Long_Charac_Descriptor . . . 102
Table 212. Aci_Gatt_Read_Long_Charac_Descriptor command parameters . . . 102
Table 213. Aci_Gatt_Read_Long_Charac_Descriptor return parameters . . . 102
Table 214. Aci_Gatt_Write_Charac_Descriptor . . . 103
Table 215. Aci_Gatt_Write_Charac_Descriptor command parameters . . . 103
Table 216. Aci_Gatt_Write_Charac_Descriptor return parameters . . . 103
Table 217. Aci_Gatt_Read_Charac_Descriptor . . . 104
Table 218. Aci_Gatt_Read_Charac_Descriptor command parameters . . . 104
Table 219. Aci_Gatt_Read_Charac_Descriptor return parameters . . . 104
Table 220. Aci_Gatt_Write_Without_Response . . . 104
Table 221. Aci_Gatt_Write_Without_Response command parameters . . . 105
Table 222. Aci_Gatt_Write_Without_Response return parameters . . . 105
Table 223. Aci_Gatt_Signed_Write_Without_Response . . . 105
Table 224. Aci_Gatt_Signed_Write_Without_Response command parameters . . . 106
Table 225. Aci_Gatt_Signed_Write_Without_Response return parameters . . . 106
Table 226. Aci_Gatt_Confirm_Indication . . . 106
Table 227. Aci_Gatt_Confirm_Indication command parameters . . . 106
Table 228. Aci_Gatt_Confirm_Indication return parameters . . . 107
Table 229. Aci_Gatt_Write_Resp . . . 107
Table 230. Aci_Gatt_Write_Resp command parameters . . . 107
Table 231. Aci_Gatt_Write_Resp return parameters . . . 108
Table 232. Aci_Gatt_Allow_Read . . . 108
Table 233. Aci_Gatt_Allow_Read command parameters . . . 108
Table 234. Aci_Gatt_Allow_Read return parameters . . . 108
Table 235. Aci_Gatt_Set_Security_Permission . . . 109
Table 236. Aci_Gatt_Set_Security_Permission command parameters . . . 109
Table 237. Aci_Gatt_Set_Security_Permission return parameters . . . 109
Table 238. Aci_Gatt_Set_Descriptor_Value . . . 110
Table 239. Aci_Gatt_Set_Descriptor_Value command parameters . . . 110
Table 240. Aci_Gatt_Set_Descriptor_Value return parameters . . . 110
Table 241. Aci_Gatt_Read_Handle_Value . . . 111
Table 242. Aci_Gatt_Read_Handle_Value command parameters . . . 111
Table 243. Aci_Gatt_Read_Handle_Value return parameters . . . 111
Table 244. GATT VS events . . . 112
Table 245. Evt_Blue_Gatt_Attribute_modified . . . 112
Table 246. Evt_Blue_Gatt_Procedure_Timeout . . . 113
Table 247. Evt_Blue_Gatt_Procedure_Complete . . . 113
Table 248. Evt_Blue_Gatt_Disc_Read_Charac_By_UUID_Resp . . . 113
Table 249. Evt_Blue_Gatt_Write_Permit_req . . . 114
Table 250. Evt_Blue_Gatt_Read_Permit_Req . . . 114
Table 251. Evt_Blue_Gatt_Read_Multi_Permit_Req . . . 115
Table 252. Other GATT VS events . . . 115
Table 253. HCI VS commands . . . 116
Table 254. Hal_IO_Init . . . 117
Table 255. Hal_IO_Init return parameters . . . 117
Table 256. Hal_IO_Configure . . . 117

Table 257. Hal_IO_Configure command parameters . . . 117
Table 258. Hal_IO_Configure return parameters . . . 117
Table 259. Hal_IO_set_bit . . . 118
Table 260. Hal_IO_set_bit command parameters . . . 118
Table 261. Hal_IO_set_bit return parameters . . . 118
Table 262. Hal_IO_get_bit . . . 118
Table 263. Hal_IO_get_bit command parameters . . . 118
Table 264. Hal_IO_get_bit return parameters . . . 119
Table 265. Write_Config_Data . . . 119
Table 266. Write_Config_Data command parameters . . . 119
Table 267. Write_Config_Data members . . . 119
Table 268. Write_Config_Data return parameters . . . 120
Table 269. Read_Config_Data . . . 120
Table 270. Read_Config_Data command parameters . . . 120
Table 271. Read_Config_Data return parameters . . . 120
Table 272. Set_Tx_power_level . . . 121
Table 273. Set_Tx_power_level command parameters . . . 121
Table 274. Tx_power_level command parameters combination . . . 121
Table 275. Set_Tx_power_level return parameters . . . 122
Table 276. Device_Standby . . . 122
Table 277. Device_Standby return parameters . . . 122
Table 278. Number_Of_Packet_Sent_in_DirectTX . . . 123
Table 279. Number_Of_Packet_Sent_in_DirectTX return parameters. . . . 123
Table 280. Tone_Start . . . 123
Table 281. Tone_Start command parameters . . . 124
Table 282. Tone_Start return parameters . . . 124
Table 283. Tone_Stop . . . 124
Table 284. Tone_Stop return parameters. . . . 124
Table 285. SPI pins . . . 125
Table 286. Document revision history . . . 130

List of figures

Figure 1. Bluetooth LE system with separate host and controller 14
Figure 2. Bluetooth LE system with combined host and controller 15
Figure 3. BlueNRG ACI architecture 15
Figure 4. HCI command packet 16
Figure 5. HCI event packet 18
Figure 6. HCI ACL data packet 18
Figure 7. OpCode format for VS Commands 21
Figure 8. Event parameter 0 format for VS events 22
Figure 9. SPI wire connection 126
Figure 10. SPI header format 127

1 Overview

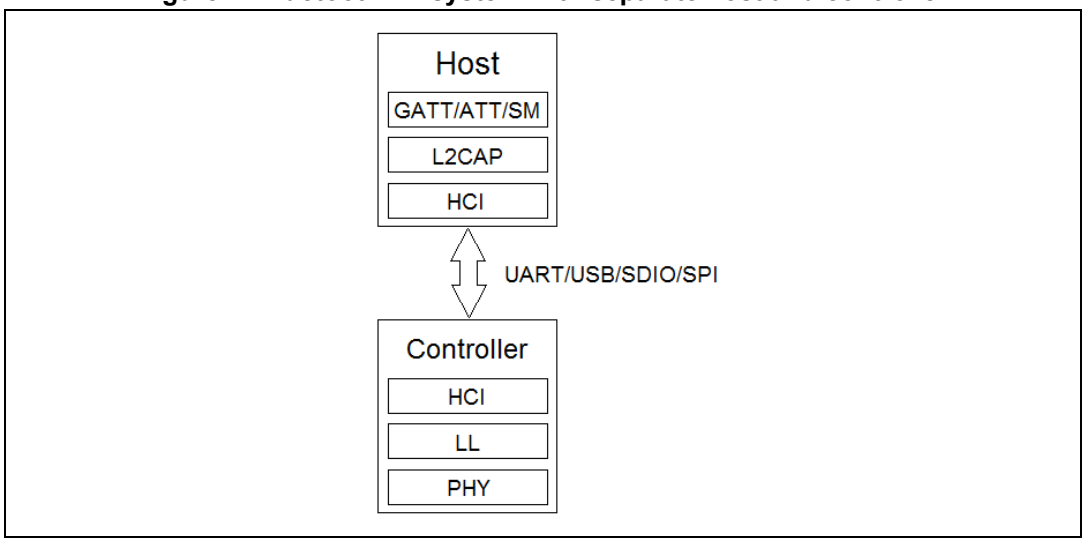
Bluetooth LE technology was adopted by the Bluetooth SIG starting from the Bluetooth core specification v4.0. Bluetooth LE technology was designed to enable products that require lower power consumption, lower complexity and lower cost compared to the Bluetooth classic or Bluetooth high-speed systems.

A typical BLE system consists of an LE controller and a host. The LE controller consists of a physical layer (PHY) including the radio, a link layer (LL) and a standard host controller interface (HCI). The host consists of a HCI and other higher protocol layers, e.g. L2CAP, SM, ATT/GATT, GAP etc.

In many designs the LE controller and the host reside in two separate silicon chips and are controlled by 2 different microcontrollers. Communication between the two is transmitted via a hardware connection, e.g. UART, SPI, USB, etc. The host can send HCI commands to control the LE controller. The HCI interface and the HCI commands are standardized by the Bluetooth core specification. Please refer to the official document for more information.

The HCI interface provides a significant benefit. Any Bluetooth tester can easily connect to the controller via the hardware connection, e.g. SPI, to test the controller by sending HCI commands. This removes the need to involve the host if the only interest is to test the controller.

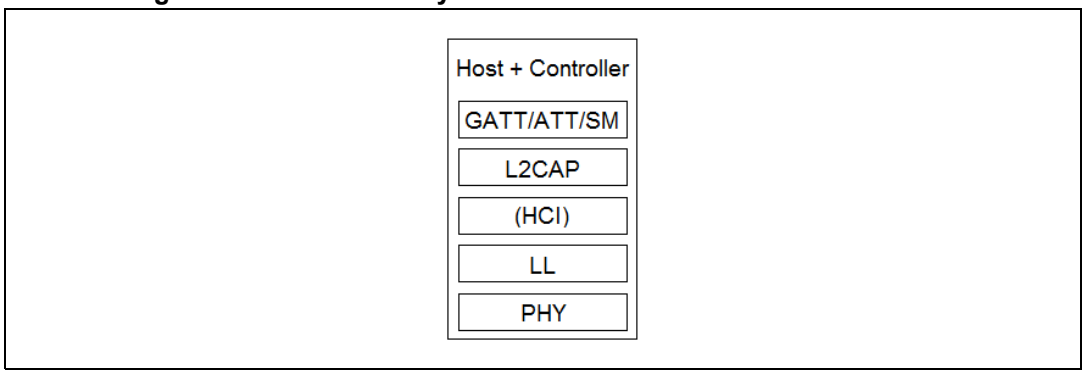
Figure 1. Bluetooth LE system with separate host and controller



Sometimes the LE controller and the host can be combined into a single chipset solution. This lowers the cost of hardware, first because it reduces the number of silicon chips to one rather than two, and second because the hardware connection between the host and the controller is no longer required.

Under this scenario, the host may directly control the LL/PHY. In this way, there is no need to generate the standard HCI commands, transmit the commands to the LL, and then process them. As a result, the execution time is improved and the calculation load on the CPU is reduced. However, if the HCI is removed completely, it will be more difficult to test only the controller with an external Bluetooth tester.

Figure 2. Bluetooth LE system with combined host and controller



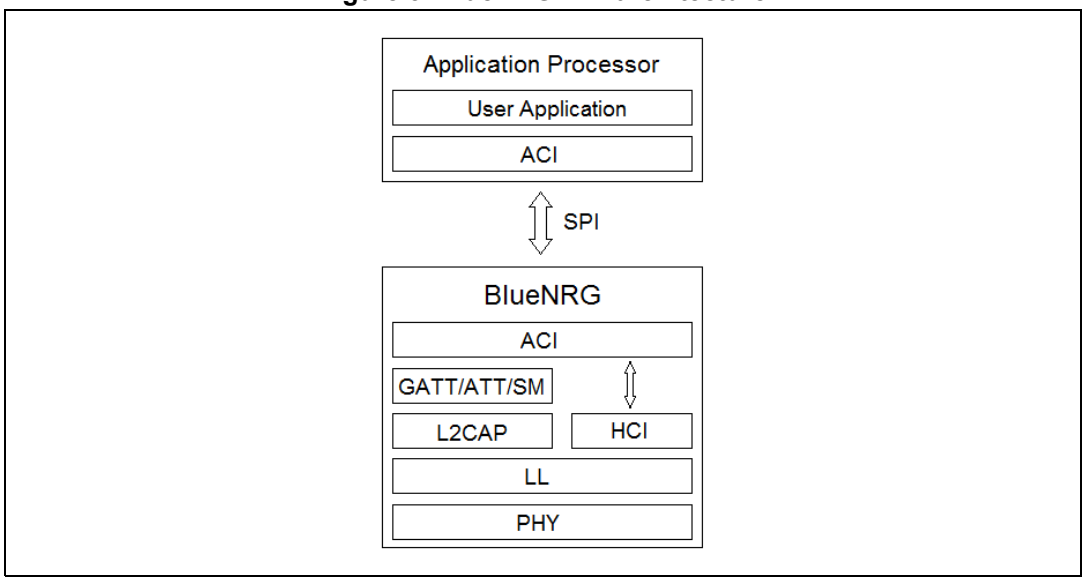
1.1 BlueNRG ACI architecture

The BlueNRG is implemented using a combined host and controller solution. An application command interface (ACI) is provided for accessing the BlueNRG host and controller.

User applications, i.e. programs running in another silicon chip, can send ACI commands to control the BlueNRG. The ACI commands should be sent over an SPI connection. The SPI protocol is described in a later section.

The ACI interface also supports HCI commands. If a command is received the ACI will check whether the command is for the host or for the controller. If the command is a HCI command, i.e. a command for the controller, the ACI will forward directly the command to the controller, bypassing it from the host. This implementation has two advantages: (1) normally the host can control the LL/PHY without using the HCI commands, which improves performance, and (2) user applications can still test the controller individually or set up some low-level hardware parameters with the HCI commands, without going through the host.

Figure 3. BlueNRG ACI architecture



2 ACI command data format

When sending an ACI command to the BlueNRG, the command must be formatted as described in this chapter.

2.1 Overview

The BlueNRG ACI commands utilize and extend the standard HCI data format. The standard HCI data format is a part of the Bluetooth core specification. To avoid redundancy and inconsistency, the texts are not copied into this document. Please refer to the official Bluetooth specification, Volume 2, Part E, Chapter 5 for detailed information.

According to the Bluetooth specification, a standard HCI packet can be an:

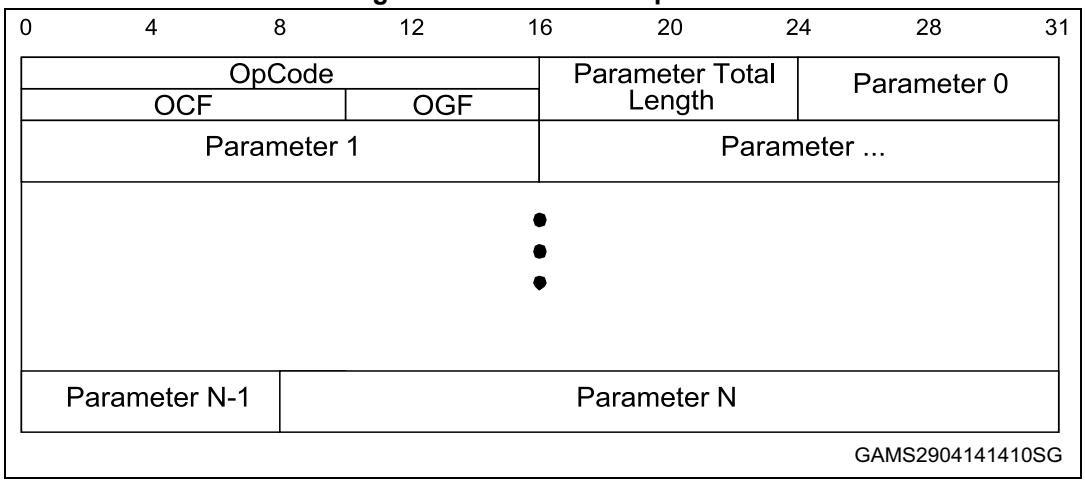
1. HCI command packet
2. HCI ACL data packet
3. HCI synchronous data packet
4. HCI event packet

In the BlueNRG, the HCI synchronous data packet is not supported, but the other three are.

2.2 HCI command packet

When an external device gives a command to the BlueNRG, the command must be formatted in a HCI command packet. The BlueNRG only receives the HCI command packets, but does not transmit.

Figure 4. HCI command packet



Each HCI command uses a 2 byte OpCode to uniquely identify different types of commands. The OpCode is divided into two fields: the OpCode Group Field (OGF) and the OpCode Command Field (OCF). The OGF is the upper 6 bits and the remaining lower 10 bits are used by the OCF.

All HCI commands are grouped into logical groups by the Bluetooth specification and each group is assigned a unique OGF value.

Table 1. OGF value

Group name	OGF value
Link Control Commands	0x01
Link Policy Commands	0x02
Controller and Baseband Commands	0x03
Information Parameters	0x04
Status Parameters	0x05
Testing Commands	0x06
LE Controller Commands	0x08
Vendor Specific Commands	0x3F

Depending on the OGF, the commands can be categorized into 3 sets:

1. Standard HCI commands: the commands with an OGF value other than 0x3F. These commands are designed for the controller. All standard HCI command are clearly defined in the Bluetooth specification, so they are not described in this document.
2. Vendor specific (VS) HCI commands: the commands with an OGF value 0x3F and are designed for the controller. Each vendor can define their own VS commands based on the hardware implementation. The VS commands defined for the BlueNRG are described in this document.
3. Vendor specific (VS) ACI commands: the commands also with an OGF value 0x3F, but these commands are designed to control/access the host. The Bluetooth specification does not define any command for the host, so all the ACI commands are naturally vendor specific. The ACI commands for the BlueNRG are described in this document.

To clarify, the commands designed to control the controller are called HCI commands. This name is defined by the Bluetooth specification and we maintain it here in order to comply with the specification. The commands designed to control the host are called ACI commands. We give it a different name to indicate that this command is rather assigned to the host, i.e. to the whole BLE system, and not only to the controller at the lower level.

However, because both ACI and HCI commands are the commands received from the external device via the ACI interface, and both of them share the same data format, sometimes it is not very important to differentiate between the two. So in this document, when referring in general to commands, we use the term “ACI commands”.

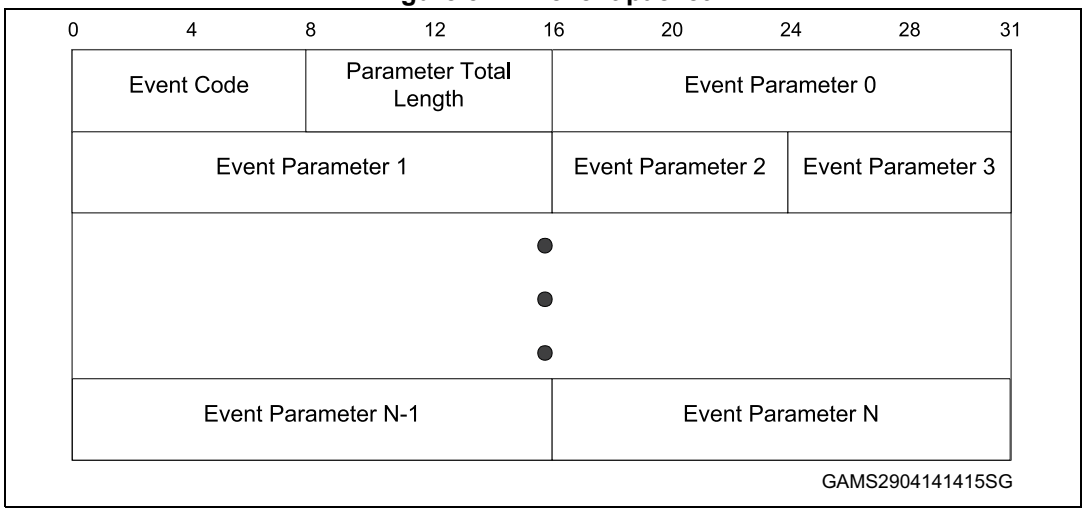
In practice, the value of the OpCode is more important. Each command, regardless of whether it is an HCI or an ACI command, matches only one OpCode value. The user application should guarantee the OpCode value is used correctly.

Please note that the Bluetooth specification uses bit-wise little endian format for the OpCode. So in *Figure 4*, the OGF field is placed to the right. But in this document, when writing an OpCode in the format of 0xXXXX, we imply the most significant bit to the left. For example, if an OpCode is 0xFE81, its top 6-bit OGF is “111111”, i.e. 0x3F, so it is a vendor specific command. And its OCF is 10-bit 0x281. For another example, if an OpCode is 0x0406, the 6-bit OGF is 0x01, and its OCF is 0x06. So this is a standard HCI command, HCI_Disconnect.

2.3 HCI event packet

The BlueNRG uses event packets to acknowledge a command or to notify that its status has updated to the user application. The BlueNRG only sends HCI event packets, but does not receive them.

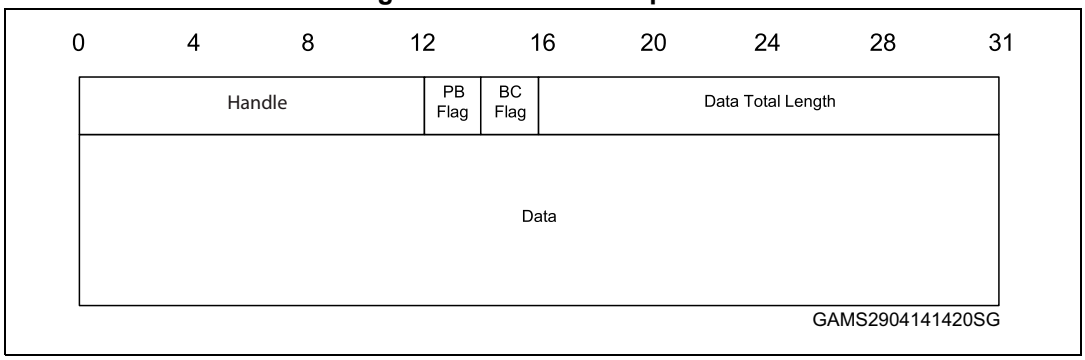
Figure 5. HCI event packet



2.4 HCI ACL data packet

The ACL data packets are used to exchange data.

Figure 6. HCI ACL data packet



3 Standard HCI commands

3.1 Supported HCI commands

The table below lists the standard Bluetooth HCI commands which are supported by the BlueNRG. For the details of each command, e.g. command descriptions, parameters, etc. please refer to the Bluetooth core specification, Volume 2, Part E, Chapter 7.

Table 2. HCI commands

Item	Command	OGF	OCF	OpCode
1	HCI_Disconnect	0x01	0x06	0x0406
2	HCI_Read_Remote_Version_Information	0x01	0x1D	0x041D
3	HCI_Set_Event_Mask	0x03	0x01	0x0C01
4	HCI_Reset	0x03	0x03	0x0C03
5	HCI_Read_Transmit_Power_Level	0x03	0x2D	0x0C2D
6	HCI_Read_Local_Version_Information	0x04	0x01	0x1001
7	HCI_Read_Local_Supported_Commands	0x04	0x02	0x1002
8	HCI_Read_Local_Supported_Features	0x04	0x03	0x1003
9	HCI_Read_BD_ADDR	0x04	0x09	0x1009
10	HCI_Read_RSSI	0x05	0x05	0x1405
11	HCI_LE_Set_Event_Mask	0x08	0x01	0x2001
12	HCI_LE_Read_Buffer_Size	0x08	0x02	0x2002
13	HCI_LE_Read_Local_Supported_Feature	0x08	0x03	0x2003
14	HCI_LE_Set_Random_Address	0x08	0x05	0x2005
15	HCI_LE_Set_Advertizing_Parameters	0x08	0x06	0x2006
16	HCI_LE_Read_Advertizing_Channel_Tx_Power	0x08	0x07	0x2007
17	HCI_LE_Set_Advertizing_Data	0x08	0x08	0x2008
18	HCI_LE_Set_Scan_Response	0x08	0x09	0x2009
19	HCI_LE_Set_Advertize_Enable	0x08	0x0A	0x200A
20	HCI_LE_Set_Scan_Parameters	0x08	0x0B	0x200B
21	HCI_LE_Set_Scan_Enable	0x08	0x0C	0x200C
22	HCI_LE_Create_Connection	0x08	0x0D	0x200D
23	HCI_LE_Create_Connection_Cancel	0x08	0x0E	0x200E
24	HCI_LE_Read_White_List_Size	0x08	0x0F	0x200F
25	HCI_LE_Clear_While_List	0x08	0x10	0x2010
26	HCI_LE_Add_Device_To_While_List	0x08	0x11	0x2011
27	HCI_LE_Remove_Device_From_While_List	0x08	0x12	0x2012
28	HCI_LE_Connection_Update	0x08	0x13	0x2013

Table 2. HCI commands (continued)

Item	Command	OGF	OCF	OpCode
29	HCI_LE_Set_Host_Channel_Classification	0x08	0x14	0x2014
30	HCI_LE_Read_Channel_Map	0x08	0x15	0x2015
31	HCI_LE_Read_Remote_Used_Features	0x08	0x16	0x2016
32	HCI_LE_Encrypt	0x08	0x17	0x2017
33	HCI_LE_Rand	0x08	0x18	0x2018
34	HCI_LE_Start_Encryption	0x08	0x19	0x2019
35	HCI_LE_Long_Term_Key_Request_Reply	0x08	0x1A	0x201A
36	HCI_LE_Long_Term_Key_Requested_Negative_Reply	0x08	0x1B	0x201B
37	HCI_LE_Read_Supported_States	0x08	0x1C	0x201C
38	HCI_LE_Receiver_Test	0x08	0x1D	0x201D
39	HCI_LE_Transmitter_Test	0x08	0x1E	0x201E
40	HCI_LE_Test_End	0x08	0x1F	0x201F

3.2 Supported HCI events

The table below lists the HCI events supported by the BlueNRG.

Table 3. HCI events

Item	Event	Event code	Sub event code
1	Disconnect_Complete	0x05	-
2	Encryption_Change_Event	0x08	-
3	Read_Remote_Version_Information_Complete	0x0C	-
4	Command_Complete_Event	0x0E	-
5	Command_Status_Event	0x0F	-
6	Hardware_Error_Event	0x10	-
7	Number_Of_Completed_Packets	0x13	-
8	Encryption_Key_Refresh_Complete	0x30	-
9	LE_Connection_Complete	0x3E	0x01
10	LE_Advertising_Report	0x3E	0x02
11	LE_Connection_Update_Complete	0x3E	0x03
12	LE_Read_Remote_Used_Features_Complete	0x3E	0x04
13	LE_Long_Term_Key_Request_Event	0x3E	0x05

4 Vendor specific commands

The VS commands can be either ACI VS commands to access the host of the BlueNRG, or the HCI VS commands to access the LE controller. Both types of the commands use the OGF value of 0x3F.

4.1 VS command and VS event format

The OCF field of the OpCode of the VS commands is further divided into two fields: Command Group ID and Command ID.

Figure 7. OpCode format for VS Commands

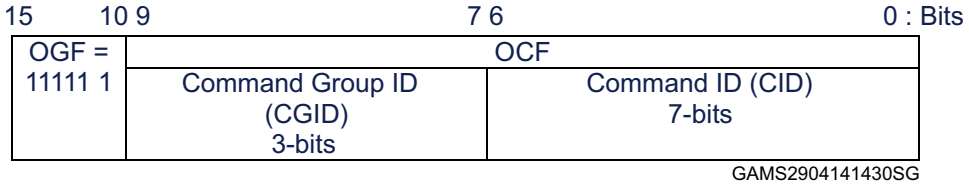


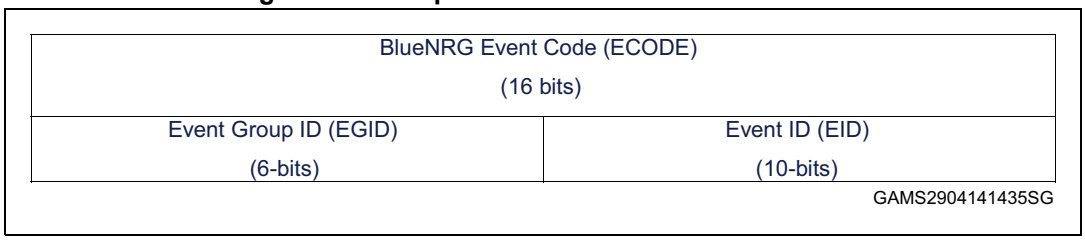
Figure 7 above gives the 16-bit OpCode format (see also *Section 2.2*). The OGF field is always 0x3F for VS commands. The 10-bit OCF field is split into two parts: 3-bit Command Group ID (CGID) and 7-bit Command ID (CID). The CGID is used by the BlueNRG ACI interface to route the commands to different logical layers, e.g. L2CAP, GAP, GATT, etc. It also helps to categorize the VS commands with a more clear structure. The CID determines the ID of each command. Each CGID group can have up to 128 VS commands.

Table 4. CGID group

Command group	Description	CGID
HCI	HCI extension commands	0x0
GAP	Generic access profile commands	0x1
GATT	Generic attribute profile commands	0x2
L2CAP	L2CAP commands	0x3
Reserved		0x4 – 0x7

The VS event also has a slightly different format than the standard HCI event (see also *Section 2.3*). (1) The 8-bit event code of the VS event always has the value of 0xFF. (2) The 16-bit event parameter 0 has a different format.

Figure 8. Event parameter 0 format for VS events



The event parameter 0 is the first return parameter in the HCI event packets, following the Parameter Length field. These 2 bytes together are defined as the BlueNRG Event Code (ECODE). ECODE is further divided into 2 fields: Event Group ID (EGIO) and Event ID (EID). BlueNRG combines the events into logical groups using the EGID. And the EID is used to specify an event in the group. The EGID occupies 6-bits in the ECODE field while the EID occupies the remaining 10 bits.

Table 5. EGID group

Event Group	Description	EGID
HCI	HCI extension events	0x0
GAP	Generic access profile commands	0x1
L2CAP	L2CAP events	0x2
GATT	Generic attribute profile events	0x3

4.2 L2CAP VS commands and events

4.2.1 L2CAP VS commands

Table 6. L2CAP VS commands

Item	Command	CGID	CID	OpCode
1	Connection Parameter Update Request	0x03	0x01	0xFD81
2	Connection Parameter Update Response	0x03	0x02	0xFD82

4.2.2 Aci_L2CAP_Connection_Parameter_Update_Request

Table 7. Aci_L2CAP_Connection_Parameter_Update_Request

Command name	Parameters	Return
Aci_L2CAP_Connection_Parameter_Update_Request (0xFD81)	Connection handle Interval min Interval max Slave latency Timeout multiplier	Status

Description:

Send an L2CAP connection parameter update request from the slave to the master.

Table 8. Aci_L2CAP_Connection_Parameter_Update_Requests command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle of the link which the connection parameter update request has to be sent
Interval min	2 bytes	Defines minimum value for the connection event interval in the following manner: connIntervalMin = Interval Min * 1.25 ms
Interval max	2 bytes	Defines maximum value for the connection event interval in the following manner: connIntervalMax = Interval Max * 1.25 ms
Slave latency	2 bytes	Defines the slave latency parameter (as number of LL connection events)
Timeout multiplier	2 bytes	Defines connection timeout parameter in the following manner: Timeout Multiplier * 10 ms

Table 9. Aci_L2CAP_Connection_Parameter_Update_Requests return parameters

Parameter	Size	Description
Status	1 byte	0x0: connection parameter accepted 0x1: connection parameter rejected

Event(s) generated:

A command status event on the receipt of the command and a Evt_Blue_L2CAP_Conn_Upd_Resp event when the master responds to the request (accepts or rejects).

4.2.3 Aci_L2CAP_Connection_Parameter_Update_Response

Table 10. Aci_L2CAP_Connection_Parameter_Update_Response

Command name	Parameters	Return
Aci_L2CAP_Connection_Parameter_Update_Response (0xFD82)	Conn_Handle Conn_Interval_Min Conn_Interval_Max Conn_Latency Timeout Identifier Accept	Status

Description:

This command should be sent in response to the Evt_Blue_L2CAP_Connection_Update_Req event from the controller. The accept parameter has to be set if the connection parameters given in the event are acceptable.

Table 11. Aci_L2CAP_Connection_Parameter_Update_Response command parameters

Parameter	Size	Description
Conn_Handle	2 bytes	Handle received in Evt_Blue_L2CAP_Connection_Update_Req event.
Conn_Interval_Min	2 bytes	Minimum connection interval received in Evt_Blue_L2CAP_Connection_Update_Req event.
Conn_Interval_Max	2 bytes	Maximum connection interval received in Evt_Blue_L2CAP_Connection_Update_Req event.
Conn_Latency	2 bytes	Connection latency received in Evt_Blue_L2CAP_Connection_Update_Req event.
Timeout	2 bytes	Supervision timeout received in Evt_Blue_L2CAP_Connection_Update_Req event.

Table 11. Aci_L2CAP_Connection_Parameter_Update_Response command parameters (continued)

Parameter	Size	Description
Identifier	1 byte	Identifier received in Evt_Blue_L2CAP_Connection_Update_Req event.
Accept	1 byte	0x00: The connection update parameters are not acceptable. 0x01: The connection update parameters are acceptable.

Table 12. Aci_L2CAP_Connection_Parameter_Update_Response return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x0C: ERR_COMMAND_DISALLOWED

Event(s) generated:

A command complete event is generated.

4.2.4 L2CAP VS events

Table 13. L2CAP VS events

Item	Event	EGID	EID	ECODE
1	Evt_Blue_L2CAP_Connection_Update_Resp	0x02	0x00	0x0800
2	Evt_Blue_L2CAP_Procedure_Timeout	0x02	0x01	0x0801
3	Evt_Blue_L2CAP_Connection_Update_Request	0x02	0x02	0x0802

4.2.5 Evt_Blue_L2CAP_Connection_Update_Resp

This event is generated when the master responds to the connection update request packet with a connection update response packet or a command reject packet.

Table 14. Evt_Blue_L2CAP_Connection_Update_Resp

Parameter	Size	Description
Event code	2 bytes	The event code for connection update response event
Connection handle	2 bytes	Connection handle for which the command is given
Datalen	1 byte	Length of the data to follow
Data	Datalen bytes	Contains: (data is in the form described in Bluetooth Core v4.0 vol 3, Part A, ch. 4.21) 1. command reject packet if the request was rejected by the master or 2. connection update response packet

4.2.6 Evt_Blue_L2CAP_Procedure_Timeout

This event is generated when the master does not respond to the connection update request packet with a connection update response packet or a command reject packet within 30 seconds.

Table 15. Evt_Blue_L2CAP_Procedure_Timeout

Parameter	Size	Description
Event code	2 bytes	The event code for L2CAP procedure timeout event
Datalen	1 byte	Length of the data to follow
Data	Datalen bytes	Status (0x0000: Success)

4.2.7 Evt_Blue_L2CAP_Connection_Update_Request

The event is given by the L2CAP layer when a connection update request is received from the slave. The upper layer which receives this event has to respond by sending a Aci_L2CAP_Connection_Update_Resp command to the BlueNRG.

Table 16. Evt_Blue_L2CAP_Connection_Update_Request

Parameter	Size	Description
Event code	2 bytes	The event code for connection update request event.
Conn_Handle	2 bytes	Handle of the connection for which the connection update request has been received. The same handle has to be returned while responding to the event with the command Aci_L2CAP_Connection_Update_Resp.
DataLen	1 byte	Length of the data to follow. The data will be the L2CAP connection update request received.
Identifier	1 byte	This is the identifier which associates the request to the response. The same identifier has to be returned by the upper layer in the command Aci_L2CAP_Connection_Update_Resp.
Length	2 bytes	Length of the L2CAP connection update request
Interval_Min	2 bytes	Value as defined in Bluetooth 4.0 spec, Volume 3, Part A 4.20
Interval_Max	2 bytes	Value as defined in Bluetooth 4.0 spec, Volume 3, Part A 4.20
Slave_Latency	2 bytes	Value as defined in Bluetooth 4.0 spec, Volume 3, Part A 4.20
Timeout_Multiplier	2 bytes	Value as defined in Bluetooth 4.0 spec, Volume 3, Part A 4.20

4.3 GAP VS commands and events

4.3.1 Overview

There are 4 GAP roles defined for LE devices:

- Broadcaster: Broadcaster is a device that sends advertising events. Broadcaster shall have a transmitter and can optionally have a receiver.
- Observer: In observer role device receives the advertising events. An observer shall have a receiver and can optionally have a transmitter.
- Peripheral: Any device that accepts the establishment of LE physical link is referred as peripheral. A device operating in peripheral role will be slave in LL connection state. A peripheral shall have both a transmitter and receiver.
- Central: A device that initiates the establishment of LE physical link is referred as central device. A device operating in central role will be master in LL connection state. A central shall have both a transmitter and receiver.

BlueNRG is capable of being the peripheral or central in the GAP profile context because:

- BlueNRG controller is capable of acting as a slave in a connection, i.e. it can accept the LL connection request from a central device and can support all the mandatory requirements of the GAP peripheral role as dictated in Table 2.1, Part C, Generic Access Profile of Bluetooth core specification 4.0.
- BlueNRG controller is capable of acting as a master in a connection, i.e. it can issue the LL connect request to a peripheral device and can support all the mandatory requirements of the GAP central role as dictated in Table 2.1, Part C, Generic Access Profile of Bluetooth core specification 4.0.

4.3.2 GAP VS Commands

Table 17. GAP VS commands

Item	Command	CGID	CID	OpCode
1	Aci_Gap_Set_Non_Discoverable	0x01	0x01	0xFC81
2	Aci_Gap_Set_Limited_Discoverable	0x01	0x02	0xFC82
3	Aci_Gap_Set_Discoverable	0x01	0x03	0xFC83
4	Aci_Gap_Set_Direct_Connectable	0x01	0x04	0xFC84
5	Aci_Gap_Set_IO_Capability	0x01	0x05	0xFC85
6	Aci_Gap_Set_Auth_Requirement	0x01	0x06	0xFC86
7	Aci_Gap_Set_Author_Requirement	0x01	0x07	0xFC87
8	Aci_Gap_Pass_Key_Response	0x01	0x08	0xFC88
9	Aci_Gap_Authorization_Response	0x01	0x09	0xFC89
10	Aci_Gap_Init	0x01	0x0A	0xFC8A
11	Aci_Gap_Set_Non_Connectable	0x01	0x0B	0xFC8B
12	Aci_Gap_Set_Undirected_Connectable	0x01	0x0C	0xFC8C
13	Aci_Gap_Slave_Security_request	0x01	0x0D	0xFC8D
14	Aci_Gap_Update_Adv_Data	0x01	0x0E	0xFC8E
15	Aci_Gap_Delete_AD_Type	0x01	0x0F	0xFC8F
16	Aci_Gap_Get_Security_Level	0x01	0x10	0xFC90
17	Aci_Gap_Set_Event_Mask	0x01	0x11	0xFC91
18	Aci_Gap_Configure_WhiteList	0x01	0x12	0xFC92
19	Aci_Gap_Terminate	0x01	0x13	0xFC93
20	Aci_Gap_Clear_Security_Database	0x01	0x14	0xFC94
21	Aci_Allow_Rebond	0x01	0x15	0xFC95
22	Aci_Gap_Start_Limited_Discovery_Proc	0x01	0x16	0xFC96
23	Aci_Gap_Start_General_Discovery_Proc	0x01	0x17	0xFC97
24	Aci_Gap_Start_Name_Discovery_Proc	0x01	0x18	0xFC98
25	Aci_Gap_Start_Auto_Conn_Establishment	0x01	0x19	0xFC99
26	Aci_Gap_Start_General_Conn_Establishment	0x01	0x1A	0xFC9A
27	Aci_Gap_Start_Selective_Conn_Establishment	0x01	0x1B	0xFC9B
28	Aci_Gap_Create_Connection	0x01	0x1C	0xFC9C
29	Aci_Gap_Terminate_Gap_Procedure	0x01	0x1D	0xFC9D
30	Aci_Gap_Start_Connection_Update	0x01	0x1E	0xFC9E
31	Aci_Gap_Send_Pairing_Request	0x01	0x1F	0xFC9F
32	Aci_Gap_Resolve_Private_Address	0x01	0x20	0xFCA0
33	Aci_Gap_Get_Bonded_Devices	0x01	0x23	0xFCA3

4.3.3 Aci_Gap_Set_Non_Discoverable

Table 18. Aci_gap_set_non_discoverable

Command name	Parameters	Return
Aci_Gap_Set_Non_Discoverable (0xFC81)		Status

Description:

Set the device in non-discoverable mode. This command will disable the LL advertising and put the device in standby state.

Table 19. Aci_gap_set_non_discoverable return parameters

Parameter	Size	Description
Status	1 byte	0x0: Status success 0xC: Command Disallowed

Event(s) generated:

When the Aci_Gap_Set_Non_Discoverable command has completed, the controller will generate a command complete event.

4.3.4 Aci_Gap_Set_Limited_Discoverable

Table 20. Aci_gap_set_limited_discoverable

Command name	Parameters	Return
Aci_Gap_Set_Limited_Discoverable (0xFC82)	Advertising_Event_Type Adv_Interval_Min Adv_Interval_Max Address_Type Adv_Filter_Policy Local_Name_Length Local_Name Service_Uuid_Length Service_Uuid_List Slave_Conn_Interval_Min Slave_Conn_Interval_Max	Status

Description:

Set the device in limited discoverable mode (as defined in GAP specification volume 3, section 9.2.3). The device will be discoverable for maximum period of TGAP (lim_adv_timeout) = 180 seconds (from errata). The advertising can be disabled at any time by issuing Aci_Gap_Set_Non_Discoverable command.

The Adv_Interval_Min and Adv_Interval_Max parameters are optional. If both are set to 0, the GAP will use default values for adv intervals for limited discoverable mode.

To allow a fast connection, the host can set Local_Name, Service_Uuid_List, Slave_Conn_Interval_Min and Slave_Conn_Interval_Max. These parameters are optional in this command. These values can be set in advertised data using GAP_Update_Adv_Data command separately.

Table 21. Aci_gap_set_limited_discoverable command parameters

Parameter	Size	Description
Advertising_Event_Type	1 byte	0x00: Connectable undirected advertising (default) 0x02: Scannable undirected advertising 0x03: Non connectable undirected advertising
Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address
Adv_Filter_Policy	1 byte	0x00: Allow scan request from any, allow connect request from any (default). 0x01: Allow scan request from white list only, allow connect request from Any. 0x02: Allow scan request from any, allow connect request from white list only. 0x03: Allow scan request from white list only, allow connect request from white list only.
Local_Name_Length	1 byte	Length of the local name string in octets. If length is set to 0x00, Local_Name parameter should not be used.
Local_Name	0-N bytes	Local name of the device. This is an ASCII string without NULL character.
Service_UUID_Length	1 byte	Length of the service UUID list in octets. If there is no service to be advertised, set this field to 0x00.
Service_UUID_List	0-N bytes	This is the list of the UUID's AD types as defined in Volume 3, Section 11.1.1 of GAP Specification
Slave_Conn_Interval_Min	2 bytes	Slave connection internal minimum value. Connection interval is defined in the following manner: connIntervalmin = Slave_Conn_Interval_Min * 1.25 ms Slave_Conn_Interval_Min range: 0x0006 to 0x0C80 Value of 0xFFFF indicates no specific minimum.
Slave_Conn_Interval_Max	2 bytes	Slave connection internal maximum value. ConnIntervalmax = Slave_Conn_Interval_Max * 1.25 ms Slave_Conn_Interval_Max range: 0x0006 to 0x0C80 Slave_ Conn_Interval_Max shall be equal to or greater than the Slave_Conn_Interval_Min. Value of 0xFFFF indicates no specific maximum.

Table 22. Aci_gap_set_limited_discoverable return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameter 0x0C: Command disallowed 0x11: Unsupported feature

Event(s) generated:

When the controller receives the command, it will generate a command status event with return parameter. Controller starts the advertising after this and when advertising timeout happens (i.e. limited discovery period has elapsed), controller generates Evt_Blue_Gap_Limited_Discoverable_Complete event.

4.3.5 Aci_Gap_Set_Discoverable

Table 23. Aci_gap_set_discoverable

Command name	Parameters	Return
Aci_Gap_Set_Discoverable (0xFC83)	Advertising_Event_Type Adv_Interval_Min Adv_Interval_Max Address_Type Adv_Filter_Policy Local_Name_Length Local_Name Service_Uuid_Length Service_Uuid_List Slave_Conn_Interval_Min Slave_Conn_Interval_Max	Status

Description:

Set the device in general discoverable mode (as defined in GAP specification volume 3, section 9.2.4). The device will be discoverable until the host issues the Aci_Gap_Set_Non_Discoverable command. The Adv_Interval_Min and Adv_Interval_Max parameters are optional. If both are set to 0, the GAP uses the default values for adv intervals for general discoverable mode.

Table 24. Aci_gap_set_discoverable command parameters

Parameter	Size	Description
Advertising_Event_Type	1 byte	0x00: Connectable undirected advertising (default) 0x02: Scannable undirected advertising 0x03: Non connectable undirected advertising
Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address

Table 24. Aci_gap_set_discoverable command parameters (continued)

Parameter	Size	Description
Adv_Filter_Policy	1 byte	0x00: Allow scan request from any, allow connect request from any (default). 0x01: Allow scan request from white list only, allow connect request from any. 0x02: Allow scan request from any, allow connect request from white list only. 0x03: Allow scan request from white list only, allow connect request from white list only.
Local_Name_Length	1 byte	Length of the local name string in octets. If length is set to 0x00, Local_Name parameter should not be used.
Local_Name	0-N bytes	Local name of the device. This is an ASCII string without NULL character.
Service_UUID_Length	1 byte	Length of the service UUID list in octets. If there is no service to be advertised, set this field to 0x00.
Service_UUID_List	0-N bytes	This is the list of the UUIDs AD types as defined in Volume 3, Section 11.1.1 of GAP Specification
Slave_Conn_Interval_Min	2 bytes	Slave connection internal minimum value. Connection interval is defined in the following manner: connIntervalmin = Slave_Conn_Interval_Min * 1.25ms Slave_Conn_Interval_Min range: 0x0006 to 0x0C80 Value of 0xFFFF indicates no specific minimum.
Slave_Conn_Interval_Max	2 bytes	Slave connection internal maximum value. ConnIntervalmax = Slave_Conn_Interval_Max * 1.25ms Slave_Conn_Interval_Max range: 0x0006 to 0x0C80 Slave_ Conn_Interval_Max shall be equal to or greater than the Slave_Conn_Interval_Min. Value of 0xFFFF indicates no specific maximum.

Table 25. Aci_gap_set_discoverable return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameter 0x0C: Command disallowed 0x11: Unsupported feature

Event(s) generated:

When the Aci_Gap_Set_Discoverable command has completed, the controller will generate a command complete event.

4.3.6 Aci_Gap_Set_Direct_Connectable

Table 26. Aci_Gap_Set_Direct_Connectable

Command name	Parameters	Return
Aci_Gap_Set_Direct_Connectable (0xFC84)	Own_Address_Type Initiator_Address_Type Initiator_Direct_Address	Status

Description:

Set the device in direct connectable mode (as defined in GAP specification Volume 3, Section 9.3.3). If the privacy is enabled, the reconnection address is used for advertising else the address of the type specified in own_address_type is used. The device will be in directed connectable mode only for 1.28 seconds. If no connection is established within this duration, the device enters non discoverable mode and advertising will have to be again enabled explicitly.

Table 27. Aci_Gap_Set_Direct_Connectable command parameters

Parameter	Size	Description
Own_Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address
Initiator_Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address
Initiator_Direct_Address	6 bytes	Initiator Bluetooth address

Table 28. Aci_Gap_Set_Direct_Connectable return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameter 0x0C: Command disallowed 0x11: Unsupported feature

Event(s) generated:

When the command has completed, the controller will generate a command complete event. The controller also generates a LE_Connection_Complete event with the status set to directed_advertising_timeout if the connection was not established and 0x00 if the connection was successfully established.

4.3.7 Aci_Gap_Set_IO_Capability

Table 29. Aci_Gap_Set_IO_Capability

Command name	Parameters	Return
Aci_Gap_Set_IO_Capability (0xFC85)	IO Capability	Status

Description:

Set the IO capabilities of the device. This command has to be given only when the device is not in a connected state.

Table 30. Aci_Gap_Set_IO_Capability command parameters

Parameter	Size	Description
IO_Capability	1 byte	0x00: Display Only 0x01: Display yes/no 0x02: Keyboard Only 0x03: No Input, no output 0x04: Keyboard display

Table 31. Aci_Gap_Set_IO_Capability return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameter 0x0C: Command disallowed

Event(s) generated:

When the command has completed, the controller will generate a command complete event.

4.3.8 Aci_Gap_Set_Auth_Requirement

Table 32. Aci_Gap_Set_Auth_Requirement

Command name	Parameters	Return
Aci_Gap_Set_Auth_Requirement (0xFC86)	MIMT mode OOB enable OOB data Min encryption key size Max encryption key size Use fixed pin Fixed pin Bonding mode	Status

Description:

Set the authentication requirements for the device. If the OOB_Enable is set to 0, the following 16 octets of OOB_Data will be ignored on reception. This command has to be given only when the device is not in a connected state.

Table 33. Aci_Gap_Set_Auth_Requirement command parameters

Parameter	Size	Description
MIMT mode	1 byte	0x00: MIMT not required 0x01: MIMT required
OOB enable	1 byte	0x00: Out Of Bound authentication not enabled 0x01: Out Of Bound authentication enabled
OOB data	16 bytes	OOB data
Min encryption key size	1 byte	Minimum size of the encryption key to be used during the pairing process
Max encryption key size	1 byte	Maximum size of the encryption key to be used during the pairing process
Use fixed pin	1 byte	0x00: Use fixed pin during the pairing process 0x01: Do not use fixed pin during the pairing process. In this case, GAP will generate Evt_Blue_Pin_Code_Request event to the host.
Fixed pin	4 bytes	Fixed pin to be used during pairing if MIMT protection is enabled. Any random value between 0 to 999999
Bonding mode	1 byte	0x00: Bonding not required 0x01: Bonding required

Table 34. Aci_Gap_Set_Auth_Requirement return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameter 0x0C: Command Disallowed 0x11: Unsupported Feature

Event(s) generated:

When the command has completed, the controller will generate a command complete event.

4.3.9 Aci_Gap_Set_Author_Requirement

Table 35. Aci_Gap_Set_Author_Requirement

Command name	Parameters	Return
Aci_Gap_Set_Author_Requirement (0xFC87)	Connection handle Authorization enable	Status

Description:

Set the authorization requirements of the device. This command has to be given only when the device is not in a connected state.

Table 36. Aci_Gap_Set_Author_Requirement command parameters

Parameter	Size	Description
Connection handle	2 bytes	Handle of the connection in case BlueNRG is configured as a master (otherwise it can be also 0).
Authorization enable	1 byte	0x00: Authorization not required (default) 0x01: Authorization required. This enables the authorization in the device and when a remote device tries to connect to GATT server, Evt_Blue_Gap_Authorization_Request event will be sent to the host in this case

Table 37. Aci_Gap_Set_Author_Requirement return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameter 0x0C: Command disallowed

Event(s) generated:

When the command has completed, the controller will generate a command complete event.

4.3.10 Aci_Gap_Pass_Key_Response

Table 38. Aci_Gap_Pass_Key_Response

Command name	Parameters	Return
Aci_Gap_Pass_Key_Response (0xFC88)	Conn_handle Pass_Key	Status

Description:

This command should be send by the host in response to Evt_Blue_Pass_Key_Request event. The command parameter contains the pass key which will be used during the pairing process.

Table 39. Aci_Gap_Pass_Key_Response command parameters

Parameter	Size	Description
Conn_handle	2 bytes	Connection handle
Pass_Key	4 byte	Pass key, value range: 0 - 999999 (decimal)

Table 40. Aci_Gap_Pass_Key_Response return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameter 0x0C: Command disallowed

Event(s) generated:

When controller receives this command, it will generate a command status event with return parameter. When paring process completes, it will generate Evt_Blue_Gap_Pairing_Complete event.

4.3.11 Aci_Gap_Authorization_Response

Table 41. Aci_Gap_Authorization_Response

Command name	Parameters	Return
Aci_Gap_Authorization_Response (0xFC89)	Authorize	Status

Description:

This command should be send by the host in response to Evt_Blue_Gap_Authorization_Request event.

Table 42. Aci_Gap_Authorization_Response command parameters

Parameter	Size	Description
Authorize	1 byte	0x01: Authorize (accept connection) 0x02: Reject (reject connection)

Table 43. Aci_Gap_Authorization_Response return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameter 0x0C: Command disallowed

Event(s) generated:

When the command has completed, the controller will generate a command complete event.

4.3.12 Aci_Gap_Init

Table 44. Aci_Gap_Init

Command name	Parameters	Return
Aci_Gap_Init (0xFC8A)	Role	Status Service handle Dev name char handle Appearance Char handle

Description:

Register the GAP service with the GATT. If the role is peripheral, then the device name characteristic and appearance characteristic are registered by default and the handles of these characteristics are returned in the event data.

Table 45. Aci_Gap_Init command parameters

Parameter	Size	Description
Role	1 byte	0x01: Peripheral 0x02: Broadcaster 0x03: Central 0x04: Observer Notes: BlueNRG currently supports only peripheral and central roles

Table 46. Aci_Gap_Init return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x47: Error
Service handle	2 bytes	Handle for the GAP service
Dev name char handle	2 bytes	Handle for the device name characteristic added to the GAP service
Appearance char handle	2 bytes	Handle for the appearance characteristic added to the GAP service

Event(s) generated:

On the completion of the command, a command complete event is generated. The status indicates success or failure. If the registration of GAP service is successful, the event data contains the handle for the GAP service registered with GATT and also the device name characteristic and appearance characteristic handles registered for the GAP service if the role is peripheral.

4.3.13 Aci_Gap_Set_Non_Connectable

Table 47. Aci_Gap_Set_Non_Connectable

Command name	Parameters	Return
Aci_Gap_Set_Non_Connectable (0xFC8B)	Advertising_Event_Type	Status

Description:

Put the device into non connectable mode

Table 48. Aci_Gap_Set_Non_Connectable command parameters

Parameter	Size	Description
Advertising_Event_Type	1 byte	0x02: Scannable undirected advertising 0x03: Non connectable undirected advertising

Table 49. Aci_Gap_Set_Non_Connectable return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameter 0x0C: Command disallowed 0x11: Unsupported feature

Event(s) generated:

A command complete event is generated upon the completion of the command.

4.3.14 Aci_Gap_Set_Undirected_Connectable

Table 50. Aci_Gap_Set_Undirected_Connectable

Command name	Parameters	Return
Aci_Gap_Set_Undirected_Connectable (0xFC8C)	Advertising filtering policy Own address type	Status

Description:

Put the device into undirected connectable mode. If privacy is enabled in the device, a resolvable private address is generated and used as the advertiser's address. If not, the address of the type specified in Own_Address_Type is used for advertising.

Table 51. Aci_Gap_Set_Undirected_Connectable command parameters

Parameter	Size	Description
Advertising_Event_Type	1 byte	0x00: Allow scan request from any, allow connect request from any (default). 0x03: Allow scan request from white list only, allow connect
Own_Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address

Table 52. Aci_Gap_Set_Undirected_Connectable return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameter 0x0C: Command disallowed 0x11: Unsupported feature

Event(s) generated:

A command complete event is generated on the completion of the command.

4.3.15 Aci_Gap_Slave_Security_Request

Table 53. Aci_Gap_Slave_Security_Request

Command name	Parameters	Return
Aci_Gap_Slave_Security_Request (0xFC8D)	Bonding MITM Protection	Status

Description:

This command has to be issued to notify the master of the security requirements of the slave.

Table 54. Aci_Gap_Slave_Security_Request command parameters

Parameter	Size	Description
Connection handle	2 bytes	Handle of the connection on which the slave security request will be sent (ignored in slave-only role)
Bonding	1 byte	0x00: No bonding 0x01: Bonding required
MITM protection	1 byte	0x00: MIMT protection not required 0x01: MIMT protection required

Table 55. Aci_Gap_Slave_Security_Request return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x1F: Out of memory 0x0C: Command disallowed

Event(s) generated:

A command status event will be generated when a valid command is received. On completion of the command, i.e. when the security request is successfully transmitted to the master, a Evt_Blue_Slave_Security_Initiated vendor specific event will be generated.

4.3.16 Aci_Gap_Update_Adv_Data

Table 56. Aci_Gap_Update_Adv_Data

Command name	Parameters	Return
Aci_Gap_Update_Adv_Data (0xFC8E)	Adv Data Length Adv Data	Status

Description:

This command can be used to update the advertising data for a particular AD type. If the AD type specified does not exist, then it is added to the advertising data. If the overall advertising data length is more than 31 octets after the update, then the command is rejected and the old data is retained.

Table 57. Aci_Gap_Update_Adv_Data command parameters

Parameter	Size	Description
Adv Data Length	1 byte	The number of bytes of data that is to follow in the advData parameter
Adv Data	0-N bytes	The advData has to be formatted as specified in Bluetooth specification (Volume 3, Part C, 11), including data length

Table 58. Aci_Gap_Update_Adv_Data return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x41: Failed

Event(s) generated:

A command complete event will be generated when the command has completed with the status indicating success or failure.

4.3.17 Aci_Gap_Delete_AD_Type

Table 59. Aci_Gap_Delete_AD_Type

Command name	Parameters	Return
Aci_Gap_Delete_AD_Type (0xFC8F)	AD Type	Status

Description:

This command can be used to delete the specified AD type from the advertisement data if present.

Table 60. Aci_Gap_Delete_AD_Type command parameters

Parameter	Size	Description
AD Type	1 byte	0x01: AD_TYPE_FLAGS 0x02: AD_TYPE_16_BIT_SERV_UUID 0x03: AD_TYPE_16_BIT_SERV_UUID_CMPLT_LIST 0x04: AD_TYPE_32_BIT_SERV_UUID 0x05: AD_TYPE_32_BIT_SERV_UUID_CMPLT_LIST 0x06: AD_TYPE_128_BIT_SERV_UUID 0x07: AD_TYPE_128_BIT_SERV_UUID_CMPLT_LIST 0x08: AD_TYPE_SHORTENED_LOCAL_NAME 0x09: AD_TYPE_COMPLETE_LOCAL_NAME 0x0A: AD_TYPE_TX_POWER_LEVEL 0x10: AD_TYPE_SEC_MGR_TK_VALUE 0x11: AD_TYPE_SEC_MGR_OOB_FLAGS 0x12: AD_TYPE_SLAVE_CONN_INTERVAL 0x14: AD_TYPE_SERV_SOLICIT_16_BIT_UUID_LIST 0x15: AD_TYPE_SERV_SOLICIT_32_BIT_UUID_LIST 0x16: AD_TYPE_SERVICE_DATA 0xFF: AD_TYPE_MANUFACTURER_SPECIFIC_DATA

Table 61. Aci_Gap_Delete_AD_Type return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x1F: AD Type not found

Event(s) generated:

A command complete event will be generated and the status indicates success or failure.

4.3.18 Aci_Gap_Get_Security_Level

Table 62. Aci_Gap_Get_Security_Level

Command name	Parameters	Return
Aci_Gap_Get_Security_Level (0xFC90)		Status MIMT protection required Bonding required OOB data present Passkey required

Description:

This command can be used to get the current security settings of the device.

Table 63. Aci_Gap_Get_Security_Level return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success
MIMT protection required	1 byte	0x00: Not required 0x01: Required
Bonding required	1 byte	0x00: Not required 0x01: Required
OOB data present	1 byte	0x00: Not present 0x01: Present
Passkey required	1 byte	0x00: Not required 0x01: Fixed pin is present which is being used 0x2: Passkey required for pairing. An event will be generated when required.

Event(s) generated:

The security parameters will be returned in a command complete event. The first byte indicates the status of the security manager (channel opened or closed).

4.3.19 Aci_Gap_Set_Event_Mask

Table 64. Aci_Gap_Set_Event_Mask

Command name	Parameters	Return
Aci_Gap_Set_Event_Mask (0xFC91)	Event mask	Status

Description:

It allows masking events from the GAP. The default configuration is all the events masked.

Table 65. Aci_Gap_Set_Event_Mask command parameters

Parameter	Size	Description
Event mask	2 bytes	0x0001: Evt_Blue_Gap_Limited_Discoverable 0x0002: Evt_Blue_Pairing_Complete 0x0004: Evt_Blue_Pass_Key_Request 0x0008: Evt_Blue_Authorization_Request 0x0010: Evt_Blue_Slave_Security_Initiated 0x0020: Evt_Blue_Gap_Bond_Lost

Table 66. Aci_Gap_Set_Event_Mask return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x12: Invalid HCI command parameter

Event(s) generated:

A command complete event is generated on the completion of the command.

4.3.20 Aci_Gap_Configure_WhiteList

Table 67. Aci_Gap_Configure_WhiteList

Command name	Parameters	Return
Aci_Gap_Configure_WhiteList (0xFC92)		Status

Description:

Configure the controller's white list with devices that are present in the security database.

Table 68. Aci_Gap_Configure_WhiteList return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x12: Invalid HCI command parameter 0x47: Error

Event(s) generated:

The controller will generate a command complete event on the completion of the command and the status indicates whether the white list was configured or not. The command will return an error if there are no devices in the database or it was unable to add to the white list.

4.3.21 Aci_Gap_Terminate

Table 69. Aci_Gap_Terminate

Command name	Parameters	Return
Aci_Gap_Terminate (0xFC93)	Conn handle Reason	Status

Description:

Command the controller to terminate the connection.

Table 70. Aci_Gap_Terminate command parameters

Parameter	Size	Description
Conn handle	2 bytes	Handle of the connection to be terminated
Reason	1 byte	Reason for requesting disconnection. The error code can be any of ones as specified for the disconnected command in the HCI specification

Table 71. Aci_Gap_Terminate return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x0C: Command disallowed

Event(s) generated:

The controller will generate a command status event when the command is received and a Disconnection Complete event will be generated when the link is disconnected.

4.3.22 Aci_Gap_Clear_Security_Database

Table 72. Aci_Gap_Clear_Security_Database

Command name	Parameters	Return
Aci_Gap_Clear_Security_Database (0xFC94)		Status

Description:

Clear the security database. All the devices in the security database will be removed.

Table 73. Aci_Gap_Clear_Security_Database return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x4B: Flash erase failed

Event(s) generated:

The controller will generate a command complete event on the completion of the command.

4.3.23 Aci_Allow_Rebond

Table 74. Aci_Allow_Rebond

Command name	Parameters	Return
Aci_Allow_Rebond (0xFC95)		Status

Description:

Allows the security manager to complete the pairing procedure and rebond with the master.

Table 75. Aci_Allow_Rebond return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success

Event(s) generated:

The controller will generate a command complete event on the completion of the command. Even if the command is given when it is not valid, success will be returned but internally it will have no effect.

4.3.24 Aci_Gap_Start_Limited_Discovery_Proc

Table 76. Aci_Gap_Start_Limited_Discovery_Proc

Command name	Parameters	Return
Aci_Gap_Start_Limited_Discovery_Proc (0xFC96)	Scan_Interval Scan_Window Own_Address_Type filterDuplicates	Status

Description:

Start the limited discovery procedure. The controller is commanded to start active scanning. When this procedure is started, only the devices in limited discoverable mode are returned to the upper layers.

Table 77. Aci_Gap_Start_Limited_Discovery_Proc command parameters

Parameter	Size	Description
Scan_Interval	2 bytes	Time interval from when the controller started its last LE scan until it begins the subsequent LE scan. The scan interval should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 msec to 10240 msec. For a number N, Time = N * 0.625 msec.
Scan_Window	2 bytes	Amount of time for the duration of the LE scan. Scan_Window shall be less than or equal to Scan_Interval. The scan window should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 msec to 10240 msec. For a number N, Time = N * 0.625 msec.

Table 77. Aci_Gap_Start_Limited_Discovery_Proc command parameters (continued)

Parameter	Size	Description
Own_Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address
filterDuplicates	1 byte	0x00: Do not filter the duplicates (default) 0x01: Filter duplicates

Table 78. Aci_Gap_Start_Limited_Discovery_Proc return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x0C: ERR_COMMAND_DISALLOWED

Event(s) generated:

A command status event is generated as soon as the command is given. If BLE_STATUS_SUCCESS is returned, the procedure is terminated when either the upper layers issue a command to terminate the procedure by issuing the command Aci_Gap_Terminate_Gap_Procedure with the procedure code set to 0x01 or a timeout happens. When the procedure is terminated due to any of the above reasons, Evt_Blue_Gap_Procedure_Complete event is returned with the procedure code set to 0x01. The device found when the procedure is ongoing is returned to the upper layers through the event Evt_Blue_Gap_Found_Device.

4.3.25 Aci_Gap_Start_General_Discovery_Proc

Table 79. Aci_Gap_Start_General_Discovery_Proc

Command name	Parameters	Return
Aci_Gap_Start_General_Discovery_Proc (0xFC97)	Scan_Interval Scan_Window Own_Address_Type filterDuplicates	Status

Description:

Start the general discovery procedure. The controller is commanded to start active scanning.

Table 80. Aci_Gap_Start_General_Discovery_Proc command parameters

Parameter	Size	Description
Scan_Interval	2 bytes	Time interval from when the controller started its last LE scan until it begins the subsequent LE scan. The scan interval should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 msec to 10240 msec. For a number N, Time = N * 0.625 msec.
Scan_Window	2 bytes	Amount of time for the duration of the LE scan. Scan_Window shall be less than or equal to Scan_Interval. The scan window should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 msec to 10240 msec. For a number N, Time = N * 0.625 msec
Own_Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address
filterDuplicates	1 byte	0x00: Do not filter the duplicates (default) 0x01: Filter duplicates

Table 81. Aci_Gap_Start_General_Discovery_Proc return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x0C: ERR_COMMAND_DISALLOWED

Event(s) generated:

A command status event is generated as soon as the command is given. If BLE_STATUS_SUCCESS is returned, the procedure is terminated when either the upper layers issue a command to terminate the procedure by issuing the command Aci_Gap_Terminate_Gap_Procedure with the procedure code set to 0x02 or a timeout happens. When the procedure is terminated due to any of the above reasons, Evt_Blue_Gap_Procedure_Complete event is returned with the procedure code set to 0x02.

The device found when the procedure is ongoing is returned to the upper layers through the event Evt_Blue_Gap_Found_Device.

4.3.26 Aci_Gap_Start_Name_Discovery_Proc

Table 82. Aci_Gap_Start_Name_Discovery_Proc

Command name	Parameters	Return
Aci_Gap_Start_Name_Discovery_Proc (0xFC98)	Scan_Interval Scan_Window Peer_Address_Type Peer_Address Own_Address_Type Conn_Interval_Min Conn_Interval_Max Conn_Latency Supervision_Timeout Conn_Len_Min Conn_Len_Max	Status

Description:

Start the name discovery procedure. A LE_Create_Connection call will be made to the controller by GAP with the initiator filter policy set to “ignore whitelist and process connectable advertising packets only for the specified device”. Once a connection is established, GATT procedure is started to read the device name characteristic. When the read is completed (successfully or unsuccessfully), a Evt_Blue_Gap_Procedure_Complete event is given to the upper layer. The event also contains the name of the device if the device name was read successfully.

Table 83. Aci_Gap_Start_Name_Discovery_Proc command parameters

Parameter	Size	Description
Scan_Interval	2 bytes	Time interval from when the controller started its last LE scan until it begins the subsequent LE scan. The scan interval should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 msec to 10240 msec. For a number N, Time = N * 0.625 msec.
Scan_Window	2 bytes	Amount of time for the duration of the LE scan. Scan_Window shall be less than or equal to Scan_Interval. The scan window should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 msec to 10240 msec. For a number N, Time = N * 0.625 msec.
Peer_Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address
Peer_Address	6 bytes	Address of the peer device with which a connection has to be established.
Own_Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address

Table 83. Aci_Gap_Start_Name_Discovery_Proc command parameters (continued)

Parameter	Size	Description
Conn_Interval_Min	2 bytes	Minimum value for the connection event interval. This shall be less than or equal to Conn_Interval_Max. Range: 0x0006 to 0x0C80 Time = N * 1.25 msec Time Range: 7.5 msec to 4 seconds
Conn_Interval_Max	2 bytes	Maximum value for the connection event interval. This shall be greater than or equal to Conn_Interval_Min. Range: 0x0006 to 0x0C80 Time = N * 1.25 msec Time Range: 7.5 msec to 4 seconds
Conn_Latency	2 bytes	Slave latency for the connection in number of connection events. Range: 0x0000 to 0x01F4
Supervision_Timeout	2 bytes	Supervision timeout for the LE Link. Range: 0x000A to 0x0C80 Time = N * 10 msec Time Range: 100 msec to 32 seconds
Conn_Len_Min	2 bytes	Minimum length of connection needed for the LE connection. Range: 0x0000 – 0xFFFF Time = N * 0.625 msec.
Conn_Len_Max	2 bytes	Maximum length of connection needed for the LE connection. Range: 0x0000 – 0xFFFF Time = N * 0.625 msec.

Table 84. Aci_Gap_Start_Name_Discovery_Proc return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x0C: ERR_COMMAND_DISALLOWED

Event(s) generated:

A command status event is generated as soon as the command is given. If BLE_STATUS_SUCCESS is returned, on completion of the procedure, a Evt_Blue_Gap_Procedure_Complete event is returned with the procedure code set to 0x04.

4.3.27 Aci_Gap_Start_Auto_Conn_Establishment

Table 85. Aci_Gap_Start_Auto_Conn_Establishment

Command name	Parameters	Return
Aci_Gap_Start_Auto_Conn_Establishment (0xFC99)	Scan_Interval Scan_Window Own_Address_Type Conn_Interval_Min Conn_Interval_Max Conn_Latency Supervision_Timeout Conn_Len_Min Conn_Len_Max Use_reconn_addr Reconn_addr Num_WhiteList_Entries Addr_Array	Status

Description:

Start the auto connection establishment procedure. The devices specified are added to the white list of the controller and a LE_Create_Connection call will be made to the controller by GAP with the initiator filter policy set to “use whitelist to determine which advertiser to connect to”. When a command is issued to terminate the procedure by upper layer, a LE_Create_Connection_Cancel call will be made to the controller by GAP.

Table 86. Aci_Gap_Start_Auto_Conn_Establishment command parameters

Parameter	Size	Description
Scan_Interval	2 bytes	Time interval from when the controller started its last LE scan until it begins the subsequent LE scan. The scan interval should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 msec to 10240 msec. For a number N, Time = N * 0.625 msec.
Scan_Window	2 bytes	Amount of time for the duration of the LE scan. Scan_Window shall be less than or equal to Scan_Interval. The scan window should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 msec to 10240 msec. For a number N, Time = N * 0.625 msec.
Own_Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address
Conn_Interval_Min	2 bytes	Minimum value for the connection event interval. This shall be less than or equal to Conn_Interval_Max. Range: 0x0006 to 0x0C80 Time = N * 1.25 msec Time Range: 7.5 msec to 4 seconds

Table 86. Aci_Gap_Start_Auto_Conn_Establishment command parameters

Parameter	Size	Description
Conn_Interval_Max	2 bytes	Maximum value for the connection event interval. This shall be greater than or equal to Conn_Interval_Min. Range: 0x0006 to 0x0C80 Time = N * 1.25 msec Time Range: 7.5 msec to 4 seconds
Conn_Latency	2 bytes	Slave latency for the connection in number of connection events. Range: 0x0000 to 0x01F4
Supervision_Timeout	2 bytes	Supervision timeout for the LE Link. Range: 0x000A to 0x0C80 Time = N * 10 msec Time Range: 100 msec to 32 seconds
Conn_Len_Min	2 bytes	Minimum length of connection needed for the LE connection. Range: 0x0000 – 0xFFFF Time = N * 0.625 msec.
Conn_Len_Max	2 bytes	Maximum length of connection needed for the LE connection. Range: 0x0000 – 0xFFFF Time = N * 0.625 msec.
Use_reconn_addr	1 byte	If 1, the provided reconnection address is used as our address during the procedure (the address has been previously notified to the application through EVT_BLUE_GAP_RECONNECTION_ADDRESS event)
Reconn_addr	6 bytes	Reconnection address used if use_reconn_addr is 1.
Num_WhiteList_Entries	1 byte	Number of devices that have to be added to the whitelist.
Addr_Array	N * 7 bytes	The addr_array will contain Num_White_List_Entries number of addresses and their address types that have to be added into the whitelist. The format of the addr_array should be address type followed by address.

Table 87. Aci_Gap_Start_Auto_Conn_Establishment return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x0C: ERR_COMMAND_DISALLOWED

Event(s) generated:

A command status event is generated as soon as the command is given. If BLE_STATUS_SUCCESS is returned, the procedure is terminated when either a connection is successfully established with one of the specified devices in the white list or the procedure is explicitly terminated by issuing the command.

Aci_Gap_Terminate_Gap_Procedure with the procedure code set to 0x08. A Evt_Blue_Gap_Procedure_Complete event is returned with the procedure code set to 0x08.

4.3.28 Aci_Gap_Start_General_Conn_Establishment

Table 88. Aci_Gap_Start_General_Conn_Establishment

Command name	Parameters	Return
Aci_Gap_Start_General_Conn_Establishment (0xFC9A)	Scan_Type Scan_Interval Scan_Window Own_Address_Type filterDuplicates Use_reconn_addr Reconn_addr	Status

Start a general connection establishment procedure. The host enables scanning in the controller with the scanner filter policy set to “accept all advertising packets” and from the scanning results, all the devices are sent to the upper layer using the event Evt_Blue_Gap_Found_Device. The upper layer then has to select one of the devices to which it wants to connect by issuing the command Aci_Gap_Create_Connection.

Table 89. Aci_Gap_Start_General_Conn_Establishment command parameters

Parameter	Size	Description
Scan_Type	1 byte	0x00: Passive scanning 0x01: Active scanning
Scan_Interval	2 bytes	Time interval from when the controller started its last LE scan until it begins the subsequent LE scan. The scan interval should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 msec to 10240 msec. For a number N, Time = N * 0.625 msec.
Scan_Window	2 bytes	Amount of time for the duration of the LE scan. Scan_Window shall be less than or equal to Scan_Interval. The scan window should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 msec to 10240 msec. For a number N, Time = N * 0.625 msec.
Own_Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address
filterDuplicates	1 byte	0x00: Do not filter the duplicates (default) 0x01: Filter duplicates
Use_reconn_addr	1 byte	If 1, the provided reconnection address is used as our address during the procedure (the address has been previously notified to the application through EVT_BLUE_GAP_RECONNECTION_ADDRESS event)
Reconn_addr	6 bytes	Reconnection address used if use_reconn_addr is 1.

Table 90. Aci_Gap_Start_General_Conn_Establishment return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x0C: ERR_COMMAND_DISALLOWED

Event(s) generated:

A command status event is generated when the command is issued. If the status is returned as BLE_STATUS_SUCCESS, then on completion of the procedure a Evt_Blue_Gap_Procedure_Completed event is generated with the procedure code set to 0x10. The procedure is terminated when a connection is established or the upper layer terminates the procedure by issuing the command Aci_Gap_Terminate_Gap_Procedure with the procedure code set to 0x10.

4.3.29 Aci_Gap_Start_Selective_Conn_Establishment

Table 91. Aci_Gap_Start_Selective_Conn_Establishment

Command name	Parameters	Return
Aci_Gap_Start_Selective_Conn_Establishment (0xFC9B)	Scan_Type Scan_Interval Scan_Window Own_Address_Type filterDuplicates Num_WhiteList_Entries Addr_Array	Status

Description:

Start a selective connection establishment procedure. The GAP adds the specified device addresses into white list and enables scanning in the controller with the scanner filter policy set to “accept packets only from devices in whitelist”. All the devices found are sent to the upper layer by the event Evt_Blue_Gap_Found_Device. The upper layer then has to select one of the devices to which it wants to connect by issuing the command Aci_Gap_Create_Connection.

Table 92. Aci_Gap_Start_Selective_Conn_Establishment command parameters

Parameter	Size	Description
Scan_Type	1 byte	0x00: Passive scanning 0x01: Active scanning
Scan_Interval	2 bytes	Time interval from when the controller started its last LE scan until it begins the subsequent LE scan. The scan interval should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 msec to 10240 msec. For a number N, Time = N * 0.625 msec.

Table 92. Aci_Gap_Start_Selective_Conn_Establishment command parameters

Parameter	Size	Description
Scan_Window	2 bytes	Amount of time for the duration of the LE scan. Scan_Window shall be less than or equal to Scan_Interval. The scan window should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 msec to 10240 msec. For a number N, Time = N * 0.625 msec.
Own_Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address
filterDuplicates	1 byte	0x00: Do not filter the duplicates (default) 0x01: Filter duplicates
Num_WhiteList_Entries	1 byte	Number of devices that have to be added to the whitelist.
Addr_Array	7 bytes	The addr_array will contain Num_White_List_Entries number of addresses and their address types that have to be added into the whitelist. The format of the addr_array should be address type followed by address.

Table 93. Aci_Gap_Start_General_Conn_Establishment return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x0C: ERR_COMMAND_DISALLOWED

Event(s) generated:

A command status event is generated when the command is issued. If the status is returned as BLE_STATUS_SUCCESS, then on completion of the procedure a Evt_Blue_Gap_Procedure_Completed event is generated with the procedure code set to 0x20. The procedure is terminated when a connection is established or the upper layer terminates the procedure by issuing the command Aci_Gap_Terminate_Gap_Procedure with the procedure code set to 0x20.

4.3.30 Aci_Gap_Create_Connection

Table 94. Aci_Gap_Create_Connection

Command name	Parameters	Return
Aci_Gap_Create_Connection (0xFC9C)	Scan_Interval Scan_Window Peer_Address_Type Peer_Address Own_Address_Type Conn_Interval_Min Conn_Interval_Max Conn_Latency Supervision_Timeout Conn_Len_Min Conn_Len_Max	Status

Description:

Start the direct connection establishment procedure. A LE_Create_Connection call will be made to the controller by GAP with the initiator filter policy set to “ignore whitelist and process connectable advertising packets only for the specified device”. The procedure can be terminated explicitly by the upper layer by issuing the command Aci_Gap_Terminate_Gap_Procedure. When a command is issued to terminate the procedure by upper layer, a LE_Create_Connection_Cancel call will be made to the controller by GAP.

Table 95. Aci_Gap_Create_Connection command parameters

Parameter	Size	Description
Scan_Interval	2 bytes	Time interval from when the controller started its last LE scan until it begins the subsequent LE scan. The scan interval should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 msec to 10240 msec. For a number N, Time = N * 0.625 msec.
Scan_Window	2 bytes	Amount of time for the duration of the LE scan. Scan_Window shall be less than or equal to Scan_Interval. The scan window should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 msec to 10240 msec. For a number N, Time = N * 0.625 msec.
Peer_Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address
Peer_Address	6 bytes	Address of the peer device with which a connection has to be established.
Own_Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address

Table 95. Aci_Gap_Create_Connection command parameters (continued)

Parameter	Size	Description
Conn_Interval_Min	2 bytes	Minimum value for the connection event interval. This shall be less than or equal to Conn_Interval_Max. Range: 0x0006 to 0x0C80 Time = N * 1.25 msec Time Range: 7.5 msec to 4 seconds
Conn_Interval_Max	2 bytes	Maximum value for the connection event interval. This shall be greater than or equal to Conn_Interval_Min. Range: 0x0006 to 0x0C80 Time = N * 1.25 msec Time Range: 7.5 msec to 4 seconds
Conn_Latency	2 bytes	Slave latency for the connection in number of connection events. Range: 0x0000 to 0x01F4
Supervision_Timeout	2 bytes	Supervision timeout for the LE Link. Range: 0x000A to 0x0C80 Time = N * 10 msec Time Range: 100 msec to 32 seconds
Conn_Len_Min	2 bytes	Minimum length of connection needed for the LE connection. Range: 0x0000 – 0xFFFF Time = N * 0.625 msec.
Conn_Len_Max	2 bytes	Maximum length of connection needed for the LE connection. Range: 0x0000 – 0xFFFF Time = N * 0.625 msec.

Table 96. Aci_Gap_Create_Connection return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x0C: ERR_COMMAND_DISALLOWED

Event(s) generated:

A command status event is generated as soon as the command is given. If BLE_STATUS_SUCCESS is returned, on termination of the procedure, a LE_Connection_Complete event is returned. The procedure can be explicitly terminated by the upper layer by issuing the command Aci_Gap_Terminate_Gap_Procedure with the procedure_code set to 0x40.

4.3.31 Aci_Gap_Terminate_Gap_Procedure

Table 97. Aci_Gap_Terminate_Gap_Procedure

Command name	Parameters	Return
Aci_Gap_Terminate_Gap_Procedure (0xFC9D)	Procedure_Code	Status

Description:

The gap procedure specified is terminated.

Table 98. Aci_Gap_Terminate_Gap_Procedure command parameters

Parameter	Size	Description
Procedure_Code	1 byte	0x01: LIMITED_DISCOVERY_PROC 0x02: GENERAL_DISCOVERY_PROC 0x04: NAME_DISCOVERY_PROC 0x08: AUTO_CONNECTION_ESTABLISHMENT_PROC 0x10: GENERAL_CONNECTION_ESTABLISHMENT_PROC 0x20: SELECTIVE_CONNECTION_ESTABLISHMENT_PROC 0x40: DIRECT_CONNECTION_ESTABLISHMENT_PROC

Table 99. Aci_Gap_Terminate_Gap_Procedure return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER

Event(s) generated:

A command complete event is generated for this command. If the command was successfully processed, the status field will have the value BLE_STATUS_SUCCESS and Evt_Blue_Gap_Procedure_Completed event is returned with the procedure code set to the corresponding procedure.

4.3.32 Aci_Gap_Start_Connection_Update

Table 100. Aci_Gap_Start_Connection_Update

Command name	Parameters	Return
Aci_Gap_Start_Connection_Update (0xFC9E)	Conn_Handle Conn_Interval_Min Conn_Interval_Max Conn_Latency Supervision_Timeout Conn_Len_Min Conn_Len_Max	Status

Description:

Start the connection update procedure. A LE_Connection_Update call is be made to the controller by GAP

Table 101. Aci_Gap_Start_Connection_Update command parameters

Parameter	Size	Description
Conn_Handle	2 bytes	Handle of the connection for which the update procedure has to be started.
Conn_Interval_Min	2 bytes	Minimum value for the connection event interval. This shall be less than or equal to Conn_Interval_Max. Range: 0x0006 to 0x0C80 Time = N * 1.25 msec Time Range: 7.5 msec to 4 seconds
Conn_Interval_Max	2 bytes	Maximum value for the connection event interval. This shall be greater than or equal to Conn_Interval_Min. Range: 0x0006 to 0x0C80 Time = N * 1.25 msec Time Range: 7.5 msec to 4 seconds
Conn_Latency	2 bytes	Slave latency for the connection in number of connection events. Range: 0x0000 to 0x01F4
Supervision_Timeout	2 bytes	Supervision timeout for the LE Link. Range: 0x000A to 0x0C80 Time = N * 10 msec Time Range: 100 msec to 32 seconds

Table 101. Aci_Gap_Start_Connection_Update command parameters (continued)

Parameter	Size	Description
Conn_Len_Min	2 bytes	Minimum length of connection needed for the LE connection. Range: 0x0000 – 0xFFFF Time = N * 0.625 msec.
Conn_Len_Max	2 bytes	Maximum length of connection needed for the LE connection. Range: 0x0000 – 0xFFFF Time = N * 0.625 msec.

Table 102. Aci_Gap_Start_Connection_Update return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x0C: ERR_COMMAND_DISALLOWED

Event(s) generated:

A command status event is generated as soon as the command is given. If BLE_STATUS_SUCCESS is returned, on completion of connection update, a LE_Connection_Update_Complete event is returned to the upper layer.

4.3.33 Aci_Gap_Send_Pairing_Request

Table 103. Aci_Gap_Send_Pairing_Request

Command name	Parameters	Return
Aci_Gap_Send_Pairing_Request (0xFC9F)	Conn_Handle Force_Rebond	Status

Description:

Send the SM pairing request to start a pairing process. The authentication requirements and IO capabilities should be set before issuing this command using the Aci_Set_IO_Capabilities and Aci_Set_Authentication_Requirements commands.

Table 104. Aci_Gap_Send_Pairing_Request command parameters

Parameter	Size	Description
Conn_Handle	2 bytes	Handle of the connection for which the pairing request has to be sent.
Force_Rebond	1 byte	0x00: Pairing request is sent only if the device has not previously bonded 0x01: Pairing request will be sent even if the device was previously bonded

Table 105. Aci_Gap_Send_Pairing_Request return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x0C: ERR_COMMAND_DISALLOWED

Event(s) generated:

A command status event is generated when the command is received. If BLE_STATUS_SUCCESS is returned in the command status event, a Evt_Blue_Pairing_Complete event is returned after the pairing process is completed.

4.3.34 Aci_Gap_Resolve_Private_Address

Table 106. Aci_Gap_Resolve_Private_Address

Command name	Parameters	Return
Aci_Gap_Resolve_Private_Address (0xFCA0)	Address	Status

Description:

This command tries to resolve the address provided with the IRKs present in its database. If the address is resolved successfully with any one of the IRKs present in the database, it returns success.

Table 107. Aci_Gap_Resolve_Private_Address command parameters

Parameter	Size	Description
Address	6 bytes	Address to be resolved.

Table 108. Aci_Gap_Resolve_Private_Address return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x48: The address was not resolved

Event(s) generated:

A command complete event is returned where the status indicates whether the command was successful.

4.3.35 Aci_Gap_Get_Bonded_Devices

Table 109. Aci_Gap_Get_Bonded_Devices

Command name	Parameters	Return
Aci_Gap_Get_Bonded_Devices(0xFCA3)		Status Num_of_Addresses Address_Type[i] Address[i]

Description: This command allows to get the list of bonded devices (address type and address for each bonded device).

Table 110. Aci_Gap_Get_Bonded_Devices return parameters

Parameter	Size	Description
Status	1 byte	0x00: success 0x64: Insufficient resources
Num_of_Addresses	1 byte	Number of bonded device
Address_Type[i]	1 byte * Num_of_Addresses	0x00: Public Device Address 0x01: Random Device Address
Address[i]	6 bytes * Num_of_Addresses	Public Device Address or Random Device Address of the device in the list

Event(s) generated:

When the Aci_Gap_Get_Bonded_Devices command is completed, controller will generate a command complete event with status, number of addresses, address type and address value of all the devices in the list.

4.4 GAP VS events

Table 111. GAP VS events

Item	Command	CGID	CID	OpCode
1	Evt_Blue_Gap_Limited_Discoverable	0x01	0x00	0x0400
2	Evt_Blue_Gap_Pairing_Complete	0x01	0x01	0x0401
3	Evt_Blue_Pass_Key_Request	0x01	0x02	0x0402
4	Evt_Blue_Authorization_Request	0x01	0x03	0x0403
5	Evt_Blue_Slave_Security_Initiated	0x01	0x04	0x0404
6	Evt_Blue_Gap_Bond_Lost	0x01	0x05	0x0405
7	Evt_Blue_Gap_Device_Found	0x01	0x06	0x0406

Table 111. GAP VS events (continued)

Item	Command	CGID	CID	OpCode
8	Evt_Blue_Gap_Procedure_Complete	0x01	0x07	0x0407
9	Evt_Blue_Gap_Reconnection_Address	0x01	0x08	0x0408

4.4.1 Evt_Blue_Gap_Limited_Discoverable

This event is generated by the controller when the limited discoverable mode ends due to timeout. The timeout is 180 seconds.

4.4.2 Evt_Blue_Gap_Pairing_Complete

This event is generated when the pairing process has completed successfully or a pairing procedure timeout has occurred or the pairing has failed. This is to notify the application that we have paired with a remote device so that it can take further actions or to notify that a timeout has occurred so that the upper layer can decide to disconnect the link.

Table 112. Evt_Blue_Gap_Pairing_Complete

Parameter	Size	Description
Event code	2 bytes	0x0401
Connection handle	2 bytes	Connection handle on which the pairing procedure completed
Status	1 byte	0x00: Pairing success Pairing with a remote device was successful 0x01: Pairing timeout The SMP timeout has elapsed and no further SMP commands will be processed until reconnection 0x02: Pairing failed The pairing failed with the remote device

4.4.3 Evt_Blue_Pass_Key_Request

This event is generated by the Security manager to the application when a passkey is required for pairing. When this event is received, the application has to respond with the Aci_Gap_Pass_Key_Response command.

The event is given by the GAP layer to the upper layers when a device is discovered during

scanning as a consequence of one of the GAP procedures started by the upper layers. This event is given for all the devices that are scannable (ADV_IND or ADV_SCAN_IND), regardless of the discoverable mode of the device. A Evt_Blue_Gap_Device_Found event must be discarded by the application if it does not follow a Evt_Blue_Gap_Device_Found event with advertising data (ADV_IND or ADV_SCAN_IND) from the same device.

Table 113. Evt_Blue_Pass_Key_Request

Parameter	Size	Description
conn_handle	2 bytes	Connection handle for which the passkey has been requested

4.4.4 Evt_Blue_Authorization_Request

This event is generated by the Security manager to the application when the application has set that authorization is required for reading/writing of attributes. This event will be generated as soon as the pairing is complete. When this event is received, Aci_Gap_Authorization_Response command should be used to respond by the application.

Table 114. Evt_Blue_Authorization_Request

Parameter	Size	Description
Event code	2 bytes	0x0403
Connection handle	2 bytes	Connection handle for which authorization is being requested

4.4.5 Evt_Blue_Slave_Security_Initiated

This event is generated when the slave security request is successfully sent to the master.

4.4.6 Evt_Blue_Gap_Bond_Lost

This event is generated when a pairing request is issued in response to a slave security request from a master which has previously bonded with the slave. When this event is received, the upper layer has to issue the command Aci_Allow_Rebond in order to allow the slave to continue the pairing process with the master.

4.4.7 Evt_Blue_Gap_Device_Found

The event is given by the GAP layer to the upper layers when a device is discovered during scanning as a consequence of one of the GAP procedures started by the upper layers. This event is given for all the devices that are scannable (ADV_IND or ADV_SCAN_IND), regardless of the discoverable mode of the device. A Evt_Blue_Gap_Device_Found event must be discarded by the application if it does not follow a Evt_Blue_Gap_Device_Found event with advertising data (ADV_IND or ADV_SCAN_IND) from the same device

Table 115. Evt_Blue_Gap_Device_Found

Parameter	Size	Description
Event_code	2 bytes	0x0406
Event_type	1 bytes	Type of event: ADV_IND: 0x00 ADV_DIRECT_IND: 0x01 ADV_SCAN_IND: 0x02 ADV_NONCONN_IND:0x03 SCAN_RSP: 0x04
Peer_Address_Type	1 byte	0x00: Public device address 0x01: Random device address
Peer_Address	6 bytes	Address of the peer device found during scanning.
Length_Data	1 byte	Length of advertising or scan response data

Table 115. Evt_Blue_Gap_Device_Found (continued)

Parameter	Size	Description
Data	Length_Data bytes	Advertising or scan response data
RSSI	1 byte	Size: 1 Octet (signed integer) Range: -127 ≤ N ≤ +20 Units: dBm 127: RSSI is not available

4.4.8 Evt_Blue_Gap_Procedure_Complete

This event is sent by the GAP to the upper layers when a procedure previously started has been terminated by the upper layer or has completed for any other reason

Table 116. Evt_Blue_Gap_Procedure_Complete

Parameter	Size	Description
Event code	2 bytes	0x0407
Procedure_Code	1 byte	0x01: LIMITED_DISCOVERY_PROC 0x02: GENERAL_DISCOVERY_PROC 0x04: NAME_DISCOVERY_PROC 0x08: AUTO_CONNECTION_ESTABLISHMENT_PROC 0x10: GENERAL_CONNECTION_ESTABLISHMENT_PROC 0x20: SELECTIVE_CONNECTION_ESTABLISHMENT_PROC 0x40: DIRECT_CONNECTION_ESTABLISHMENT_PROC
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x41: BLE_STATUS_FAILED 0x05: ERR_AUTH_FAILURE, Procedure failed due to authentication requirements
Procedure_Specific_Data	1 or more bytes	Procedure Aci_Gap_Name_Discovery_Proc: The name of the peer device if the procedure completed successfully Procedure Aci_Gap_General_Conn_Establishment_Proc: The reconnection address written to the peripheral device if the peripheral is privacy enabled

4.4.9 Evt_Blue_Gap_Reconnection_Address

This event is generated when the reconnection address is generated during the general connection establishment procedure. The same address is set to the peer device also as a part of the general connection establishment procedure. In order to make use of the reconnection address the next time while connecting to the bonded peripheral, the application needs to set its own address as well as the peer address to which it wants to connect to this reconnection address.

Table 117. Evt_Blue_Gap_Reconnection_Address

Parameter	Size	Description
Event code	2 bytes	0x0408
Reconnection address	6 bytes	6 bytes of reconnection address

4.5 GATT VS commands and events

4.5.1 GATT VS commands

Table 118. GATT VS commands

Item	Command	CGID	CID	OpCode
1	Aci_Gatt_Init	0x02	0x01	0xFD01
2	Aci_Gatt_Add_Service	0x02	0x02	0xFD02
3	Aci_Gatt_Include_Service	0x02	0x03	0xFD03
4	Aci_Gatt_Add_Characteristic	0x02	0x04	0xFD04
5	Aci_Gatt_Add_Char_Desc	0x02	0x05	0xFD05
6	Aci_Gatt_Update_Char_Value	0x02	0x06	0xFD06
7	Aci_Gatt_Del_Characteristic	0x02	0x07	0xFD07
8	Aci_Gatt_Del_Service	0x02	0x08	0xFD08
9	Aci_Gatt_Del_Include_Service	0x02	0x09	0xFD09
10	Aci_Gatt_Set_Event_Mask	0x02	0x0A	0xFD0A
11	Aci_Gatt_Exchange_configuration	0x02	0x0B	0xFD0B
12	Aci_Att_Find_Information_Req	0x02	0x0C	0xFD0C
13	Aci_Att_Find_Attr_By_Typ_And_Value_Req	0x02	0x0D	0xFD0D
14	Aci_Att_Find_Attr_By_Typ_And_Value_Req	0x02	0x0E	0xFD0E
15	Aci_Att_Read_Attr_By_Grp_Type_Req	0x02	0x0F	0xFD0F
16	Aci_Att_Prepare_Write_Req	0x02	0x10	0xFD10
17	Aci_Att_Execute_Write_Req	0x02	0x11	0xFD11
18	Aci_Gatt_Disc_All_Prim_Services	0x02	0x12	0xFD12
19	Aci_Gatt_Disc_Prim_serv_By_Uuid	0x02	0x13	0xFD13

Table 118. GATT VS commands (continued)

Item	Command	CGID	CID	OpCode
20	Aci_Gatt_Find_Included_Services	0x02	0x14	0xFD14
21	Aci_Gatt_Disc_All_Charac_Of_Serv	0x02	0x15	0xFD15
22	Aci_Gatt_Disc_Charac_By_UUID	0x02	0x16	0xFD16
23	Aci_Gatt_Disc_All_Charac_Descriptors	0x02	0x17	0xFD17
24	Aci_Gatt_Read_Charac_Val	0x02	0x18	0xFD18
25	Aci_Gatt_Read_Charac_Using_UUID	0x02	0x19	0xFD19
26	Aci_Gatt_Read_Long_Charac_Val	0x02	0x1A	0xFD1A
27	Aci_Gatt_Read_Multiple_Charac_Values	0x02	0x1B	0xFD1B
28	Aci_Gatt_Write_Charac_Val	0x02	0x1C	0xFD1C
29	Aci_Gatt_Write_Long_Charac_Val	0x02	0x1D	0xFD1D
30	Aci_Gatt_Write_Charac_Reliable	0x02	0x1E	0xFD1E
31	Aci_Gatt_Write_Long_Charac_Descriptor	0x02	0x1F	0xFD1F
32	Aci_Gatt_Read_Long_Charac_Descriptor	0x02	0x20	0xFD20
33	Aci_Gatt_Write_Charac_Descriptor	0x02	0x21	0xFD21
34	Aci_Gatt_Read_Charac_Descriptor	0x02	0x22	0xFD22
35	Aci_Gatt_Write_Without_Response	0x02	0x23	0xFD23
36	Aci_Gatt_Signed_Write_Without_Response	0x02	0x24	0xFD24
37	Aci_Gatt_Confirm_Indication	0x02	0x25	0xFD25
38	Aci_Gatt_Write_Resp	0x02	0x26	0xFD26
39	Aci_Gatt_Allow_Read	0x02	0x27	0xFD27
40	Aci_Gatt_Set_Security_Permission	0x02	0x28	0xFD28
41	Aci_Gatt_Set_Descriptor_Value	0x02	0x29	0xFD29
42	Aci_Gatt_Read_Handle_Value	0x02	0x2A	0xFD2A

4.5.2 Aci_Gatt_Init

Table 119. Aci_Gatt_Init

Command name	Parameters	Return
Aci_Gatt_Init (0xFD01)		Status

Description:

Initialize the GATT server on the slave device. Initialize all the pools and active nodes. Also it adds GATT service with service changed characteristic. Until this command is issued the GATT channel will not process any commands even if the connection is opened. This command has to be given before using any of the GAP features.

Table 120. Aci_Gatt_Init return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x47: Error

4.5.3 Aci_Gatt_Add_Service

Table 121. Aci_Gatt_Add_Service

Command name	Parameters	Return
Aci_Gatt_Add_Service (0xFD02)	Service_UUID_Type Service_UUID Service_Type Max_Attribute_Records	Status Service handle

Description:

Add a service to GATT Server. When a service is created in the server, the host needs to reserve the handle ranges for this service using Max_Attribute_Records parameter. This parameter specifies the maximum number of attribute records that can be added to this service (including the service attribute, include attribute, characteristic attribute, characteristic value attribute and characteristic descriptor attribute). Handle of the created service is returned in command complete event.

Table 122. Aci_Gatt_Add_Service command parameters

Parameter	Size	Description
Service_UUID_Type	1 byte	0x01: 16-bit UUID 0x02: 128-bit UUID
Service_UUID	2 bytes or 16 bytes	16-bit or 128-bit UUID based on the UUID type field

Table 122. Aci_Gatt_Add_Service command parameters (continued)

Parameter	Size	Description
Service_Type	1 byte	0x01: Primary service 0x02: Secondary service
Max_Attribute_ Records	1 byte	Maximum number of attribute records that can be added to this service

Table 123. Aci_Gatt_Add_Service return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x47: Error 0x1F: Out of memory 0x61: Invalid parameter 0x62: Out of handle 0x64: Insufficient resources
Service handle	2 bytes	Handle of the Service When this service is added to the service, a handle is allocated by the server to this service. Also server allocates a range of handles for this service from Service_Handle to (Service_Handle + Max_Attribute_Records)

Event(s) generated:

When the Aci_Gatt_Add_Service command is completed, controller will generate a command complete event with status and handle for the service as parameters.

4.5.4 Aci_Gatt_Include_Service

Table 124. Aci_Gatt_Include_Service

Command name	Parameters	Return
Aci_Gatt_Include_Service (0xFD03)	Service_Handle Include_Start_Handle Include_End_Handle Included_Uuid_Type Included_Uuid	Status Included handle

Description:

Include a service given by Service_Include_Handle to another service given by Service_Handle. Attribute server creates an INCLUDE definition attribute and return the handle of this in command complete event as Included_Handle.

Table 125. Aci_Gatt_Include_Service command parameters

Parameter	Size	Description
Service Handle	2 bytes	Handle of the service to which another service has to be included
Include_Start_Handle	2 bytes	Start handle of the service which has to be included in service
Include_End_Handle	2 bytes	End handle of the service which has to be included in service
Include_UUID_Type	1 byte	0x01: 16-bit UUID 0x02: 128-bit UUID
Include_UUID	2 bytes or 16 bytes	16-bit or 128-bit UUID of the service

Table 126. Aci_Gatt_Include_Service return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x47: Error 0x1F: Out of Memory 0x60: Invalid handle 0x61: Invalid parameter 0x62: Out of handle 0x63: Invalid Operation 0x64: Insufficient resources
Included handle	2 bytes	Handle of the included declaration

Event(s) generated:

When the Aci_Gatt_Add_Service command is completed, controller will generate a command complete event with status and handle for the service as parameters.

4.5.5 Aci_Gatt_Add_Characteristic

Table 127. Aci_Gatt_Add_Characteristic

Command name	Parameters	Return
Aci_Gatt_Add_Characteristic (0xFD04)	Service_Handle Char_UUID_Type Char_UUID Char_Value_Length Char_Properties Security_Permissions Gatt_Evt_Mask Encryption_Key_Size isVariable	Status Char handle

Description:

Add a characteristic to a service.

Table 128. Aci_Gatt_Add_Characteristic command parameters

Parameter	Size	Description
Service_Handle	2 bytes	Handle of the service to which the characteristic has to be added
Char_UUID_Type	1 byte	0x01: 16-bit UUID 0x02: 128-bit UUID
Char_UUID	2 bytes or 16 bytes	16-bit or 128-bit UUID
Char_Value_Length	1 byte	Maximum length of the characteristic value
Char_Properties	1 byte	Bitwise OR values of Characteristic Properties (defined in Volume 3, Section 3.3.3.1 of Bluetooth Specification 4.0)
Security_Permissions	1 byte	0x00: ATTR_PERMISSION_NONE 0x01: Need authentication to read 0x02: Need authorization to read 0x04: Link should be encrypted to read 0x08: Need authentication to write 0x10: Need authorization to write 0x20: Link should be encrypted for write
Gatt_Evt_Mask	1 byte	0x01: GATT_SERVER_ATTR_WRITE The application will be notified when a client writes to this attribute 0x02: GATT_INTIMATE_AND_WAIT_FOR_APPL_AUTH The application will be notified when a write request/write cmd/signed write cmd is received by the server for this attribute 0x04: GATT_INTIMATE_APPL_WHEN_READ_N_WAIT The application will be notified when a read request of any type is got for this attribute
Encryption_Key_Size	1 byte	The minimum encryption key size requirement for this attribute. Valid Range: 7 to 16
isVariable	1 byte	0x00: The attribute has a fixed length value field 0x01: The attribute has a variable length value field

Table 129. Aci_Gatt_Add_Characteristic return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x47: Error 0x1F: Out of Memory 0x60: Invalid handle 0x61: Invalid parameter 0x62: Out of handle 0x64: Insufficient resources 0x66: Character Already Exists
Char_Handle	2 bytes	Handle of the Characteristic

Event(s) generated:

When the command is completed, a command complete event will be generated by the controller which carries the status of the command and the handle of the characteristic as parameters.

4.5.6 Aci_Gatt_Add_Char_Desc

Table 130. Aci_Gatt_Add_Char_Desc

Command name	Parameters	Return
Aci_Gatt_Add_Char_Desc (0xFD05)	Service_Handle Char_Handle Char_Desc_UUID_Type Char_Desc_UUID Char_Desc_Value_Max_Length Char_Desc_Value_Length Char_Desc_Value Security_Permissions Access_Permissions Gatt_Evt_Mask Encryption_Key_Size isVariable	Status Char desc handle

Description:

Add a characteristic descriptor to a service.

Table 131. Aci_Gatt_Add_Char_Desc command parameters

Parameter	Size	Description
Service_Handle	2 bytes	Handle of the service to which characteristic belongs
Char_Handle	2 bytes	Handle of the characteristic to which description is to be added.
Char_Desc_UUID type	1 byte	0x01: 16-bit UUID (default) 0x02: 128-bit UUID
Char_Desc_UUID	2 bytes or 16 bytes	0x2900: Characteristic Extended Properties Descriptor 0x2901: Characteristic User Descriptor 0x2902: Client configuration descriptor 0x2903: Server configuration descriptor 0x2904: Characteristic presentation format 0x2905: Characteristic aggregated format 0xXXXX: 16/128 bit user specific descriptor
Char_Desc_Value_Max_Length	1 byte	The maximum length of the descriptor value
Char_Desc_Value_Length	1 byte	Current Length of the characteristic descriptor value
Char_Desc_Value	0-N Bytes	Value of the characteristic description
Security_Permissions	1 byte	0x00: No security permission 0x01: Authentication required 0x02: Authorization required 0x04: Encryption required
Access_Permissions	1 byte	0x00: No Access 0x01: Readable 0x02: Writable 0x03: Read/Write
Gatt_Event_Mask	1 byte	0x01: GATT_SERVER_ATTR_WRITE The application will be notified when a client writes to this attribute 0x02: GATT_INTIMATE_AND_WAIT_FOR_APPL_AUTH The application will be notified when a write request/write cmd/signed write cmd is received by the server for this attribute 0x04: GATT_INTIMATE_APPL_WHEN_READ_N_WAIT The application will be notified when a read request of any type is got for this attribute
Encryption_Key_Size	1 byte	The minimum encryption key size requirement for this attribute. Valid Range: 7 to 16
isVariable	1 byte	0x00: The attribute has a fixed length value field 0x01: The attribute has a variable length value field

Table 132. Aci_Gatt_Add_Char_Desc return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x47: Error 0x1F: Out of Memory 0x60: Invalid handle 0x61: Invalid parameter 0x62: Out of handle 0x63: Invalid Operation 0x64: Insufficient resources
Char_Desc_Handle	2 bytes	Handle of the characteristic descriptor

Event(s) generated:

When this command is completed, a command complete event will be generated by the controller which carries the status of the command and the handle of the characteristic descriptor.

4.5.7 Aci_Gatt_Update_Char_Value

Table 133. Aci_Gatt_Update_Char_Value

Command name	Parameters	Return
Aci_Gatt_Update_Char_Value (0xFD06)	Serv_Handle Char_Handle Val_Offset Char_Value_Length Char_Value	Status

Description:

Update a characteristic value in a service.

Table 134. Aci_Gatt_Update_Char_Value command parameters

Parameter	Size	Description
Serv_Handle	2 bytes	Handle of the service to which characteristic belongs
Char_Handle	2 bytes	Handle of the characteristic
Val_Offset	1 byte	The offset from which the attribute value has to be updated. If this is set to 0, and the attribute value is of variable length, then the length of the attribute will be set to the Char_Value_Length. If the Val_Offset is set to a value greater than 0, then the length of the attribute will be set to the maximum length as specified for the attribute while adding the characteristic.

Table 134. Aci_Gatt_Update_Char_Value command parameters (continued)

Parameter	Size	Description
Char_Value_Length	1 byte	Length of the characteristic value in octets
Char_Value	0-N bytes	Characteristic value

Table 135. Aci_Gatt_Update_Char_Value return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x47: Error 0x60: Invalid handle 0x61: Invalid parameter 0x64: Insufficient resources

Event(s) generated:

When the command has completed, the controller will generate a command complete event.

4.5.8 Aci_Gatt_Del_Characteristic

Table 136. Aci_Gatt_Del_Characteristic

Command name	Parameters	Return
Aci_Gatt_Del_Characteristic (0xFD07)	Serv_Handle Char_Handle	Status

Description:

Delete the characteristic specified from the service.

Table 137. Aci_Gatt_Del_Characteristic command parameters

Parameter	Size	Description
Serv_Handle	2 bytes	Handle of the service to which characteristic belongs
Char_Handle	2 bytes	Handle of the characteristic to description to be deleted

Table 138. Aci_Gatt_Del_Characteristic return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x47: Error 0x60: Invalid handle 0x61: Invalid parameter 0x64: Insufficient resources

Event(s) generated:

When the Aci_Gatt_Del_Characteristic command has completed, the controller will generate a command complete event.

4.5.9 Aci_Gatt_Del_Service

Table 139. Aci_Gatt_Del_Service

Command name	Parameters	Return
Aci_Gatt_Del_Service (0xFD08)	Service handle	Status

Description:

Delete the service specified from the GATT server database.

Table 140. Aci_Gatt_Del_Service command parameters

Parameter	Size	Description
Service_Handle	2 bytes	Handle of the service to be deleted

Table 141. Aci_Gatt_Del_Service return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x47: Error 0x60: Invalid handle 0x61: Invalid parameter 0x64: Insufficient resources

Event(s) generated:

When the command has completed, the controller will generate a command complete event.

4.5.10 Aci_Gatt_Del_Include_Service

Table 142. Aci_Gatt_Del_Include_Service

Command name	Parameters	Return
Aci_Gatt_Del_Include_Service	Serv_Handle Include_Handle	Status

Description:

Delete the Include definition from the service.

Table 143. Aci_Gatt_Del_Include_Service command parameters

Parameter	Size	Description
Serv_Handle	2 bytes	Handle of the service to which Include definition belongs
Include_Handle	2 bytes	Handle of the Included definition to be deleted

Table 144. Aci_Gatt_Del_Include_Service return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x47: Error 0x60: Invalid handle 0x61: Invalid parameter 0x64: Insufficient resources

Event(s) generated:

When the command has completed, the controller will generate a command complete event.

4.5.11 Aci_Gatt_Set_Event_Mask

Table 145. Aci_Gatt_Set_Event_Mask

Command name	Parameters	Return
Aci_Gatt_Set_Event_Mask (0xFD0A)	Event mask	Status

Description:

It allows masking events from the GATT. The default configuration is all the events masked.

Table 146. Aci_Gatt_Set_Event_Mask command parameters

Parameter	Size	Description
Event mask	4 bytes	0x00000001: Evt_Blue_Gatt_Attribute_modified 0x00000002: Evt_Blue_Gatt_Procedure_Timeout 0x00000004: Evt_Blue_Att_Exchange_MTU_Resp 0x00000008: Evt_Blue_Att_Find_Information_Resp 0x00000010: Evt_Blue_Att_Find_By_Type_Value_Resp 0x00000020: Evt_Blue_Att_Read_By_Type_Resp 0x00000040: Evt_Blue_Att_Read_Resp 0x00000080: Evt_Blue_Att_Read_Blob_Resp 0x00000100: Evt_Blue_Att_Read_Multiple_Resp 0x00000200: Evt_Blue_Att_Read_By_Group_Resp 0x00000400: Evt_Blue_Att_Write_Resp 0x00000800: Evt_Blue_Att_Prepare_Write_Resp 0x00001000: Evt_Blue_Att_Exec_Write_Resp 0x00002000: Evt_Blue_Gatt_Indication 0x00004000: Evt_Blue_Gatt_notification 0x00008000: Evt_Blue_Gatt_Error_Resp 0x00010000: Evt_Blue_Gatt_Procedure_Complete

Table 147. Aci_Gatt_Set_Event_Mask return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x12: Invalid HCI Command Parameters

Event(s) generated:

A command complete event is generated on the completion of the command.

4.5.12 Aci_Gatt_Exchange_Configuration

Table 148. Aci_Gatt_Exchange_Configuration

Command name	Parameters	Return
Aci_Gatt_Exchange_Configuration (0xFD0B)	Connection handle	Status

Description:

Perform an ATT MTU exchange.

Table 149. Aci_Gatt_Exchange_Configuration command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is givens

Table 150. Aci_Gatt_Exchange_Configuration return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the ATT MTU exchange procedure is completed, a Evt_Blue_Att_Exchange_MTU_Resp event is generated. Also procedure complete event is generated to indicate end of procedure

4.5.13 Aci_Att_Find_Information_Req

Table 151. Aci_Att_Find_Information_Req

Command name	Parameters	Return
Aci_Att_Find_Information_Req (0xFD0C)	Connection handle Start handle End handle	Status

Description:

Post the Find information request.

Table 152. Aci_Att_Find_Information_Req command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
Start handle	2 bytes	Starting handle of the range of attributes to be discovered on the server
End handle	2 bytes	Ending handle of the range of attributes to be discovered on the server

Table 153. Aci_Att_Find_Information_Req return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. The responses of the procedure are given through the Evt_Blue_ Att_Find_Information_Resp event. The end of the procedure is indicated by a Evt_Blue_Procedure_Complete event.

4.5.14 Att_Find_By_Type_Value_Req

Table 154. Att_Find_By_Type_Value_Req

Command name	Parameters	Return
Att_Find_By_Type_Value_Req (0xFD0D)	Connection handle Start handle End handle UUID AttrValLen AttrVal	Status

Description:

Post the Find by type value request.

Table 155. Aci_Att_Find_Attr_By_Typ_And_Value_Req command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
Start handle	2 bytes	Starting handle of the range of attributes to be discovered on the server
End handle	2 bytes	Ending handle of the range of attributes to be discovered on the server
UUID	2 bytes	16-bit UUID

Table 155. Aci_Att_Find_Attr_By_Typ_And_Value_Req command parameters

Parameter	Size	Description
AttrValLen	1 byte	Length of the attribute value to follow. Note: Though the max attribute value that is allowed according to the spec is 512 octets, due to the limitation of the transport layer (command packet max length is 255 bytes) the value is limited to 255 bytes
AttrVal	0-N bytes	Value of the attribute to be matched

Table 156. Aci_Att_Find_Attr_By_Typ_And_Value_Req return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. The responses of the procedure are given through the Evt_Blue_ Att_Find_By_Type_Value_Resp event. The end of the procedure is indicated by a Evt_Blue_Procedure_Complete event.

4.5.15 Aci_Att_Read_Attr_By_Type_Req

Table 157. Aci_Att_Read_Attr_By_Type_Req

Command name	Parameters	Return
Aci_Att_Read_Attr_By_Type_Req (0xFD0E)	Connection handle Start handle End handle UUID type UUID	Status

Description:

It posts the read by type request.

Table 158. Aci_Att_Read_Attr_By_Type_Req command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
Start handle	2 bytes	Start handle of the range of values to be read on the server
End handle	2 bytes	End handle of the range of values to be read on the server
UUID type	1 byte	0x01: 16-bit UUID 0x02: 128-bit UUID
UUID	2 bytes or 16 bytes	UUID

Table 159. Aci_Att_Find_Attr_By_Typ_And_Value_Req return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. The responses of the procedure are given through the Evt_Blue_Att_Read_By_Type_Resp event. The end of the procedure is indicated by a Evt_Blue_Procedure_Complete event.

4.5.16 Aci_Att_Read_By_Group_Type_Req

Table 160. Aci_Att_Read_By_Group_Type_Req

Command name	Parameters	Return
Aci_Att_Read_By_Group_Type_Req (0xFD0F)	Connection handle Start handle End handle UUID type UUID	Status

Description:

It posts the read by group type request.

Table 161. Aci_Att_Read_Attr_By_Grp_Type_Req command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
Start handle	2 bytes	Start handle of the range of values to be read on the server
End handle	2 bytes	End handle of the range of values to be read on the server
UUID type	1 byte	0x01: 16-bit UUID 0x02: 128-bit UUID
UUID	2 bytes or 16 bytes	UUID

Table 162. Aci_Att_Read_Attr_By_Grp_Type_Req return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. The responses of the procedure are given through the Evt_Blue_Att_Read_By_Group_Resp event. The end of the procedure is indicated by a Evt_Blue_Procedure_Complete event.

4.5.17 Aci_Att_Prepare_Write_Req

Table 163. Aci_Att_Prepare_Write_Req

Command name	Parameters	Return
Aci_Att_Prepare_Write_Req (0xFD10)	Connection handle AttrHandle valOffset AttrValLen AttrVal	Status

Description:

This will prepare a write of the attribute value specified.

Table 164. Aci_Att_Prepare_Write_Req command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
AttrHandle	2 bytes	Handle of the attribute whose value has to be written
valOffset	2 bytes	The offset at which value has to be written
AttrValLen	1 byte	Length of the attribute value to follow
AttrVal	0-N bytes	Value of the attribute to be written

Table 165. Aci_Att_Prepare_Write_Req return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. The responses of the procedure are given through the Evt_Blue_Att_Prepare_Write_Resp event. The end of the procedure is indicated by a Evt_Blue_Procedure_Complete event.

4.5.18 Aci_Att_Execute_Write_Req

Table 166. Aci_Gatt_Execute_Write_Req

Command name	Parameters	Return
Aci_Att_Execute_Write_Req (0xFD11)	Connection handle Execute	Status

Description:

All the writes previously prepared will be executed. They will be either written or discarded as specified by the execute parameter.

Table 167. Aci_Att_Execute_Write_Req command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
Execute	1 byte	TRUE: Execute the pending prepared writes FALSE: Cancel all pending writes that are prepared

Table 168. Aci_Att_Execute_Write_Req return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If ATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. The result of the procedure is given through the Evt_Blue_Att_Exec_Write_Resp event. The end of the procedure is indicated by a Evt_Blue_Procedure_Complete event.

4.5.19 Aci_Gatt_Disc_All_Prim_Services

Table 169. Aci_Gatt_Disc_All_Prim_Services

Command name	Parameters	Return
Aci_Gatt_Disc_All_Prim_Services (0xFD12)	Connection handle	Status

Description:

This command will start the GATT client procedure to discover all primary services on the server.

Table 170. Aci_Gatt_Disc_All_Prim_Services command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given

Table 171. Aci_Gatt_Disc_All_Prim_Services return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. The responses of the procedure are given through the Evt_Blue_Att_Read_By_Group_Resp event. The end of the procedure is indicated by a Evt_Blue_Procedure_Complete event.

4.5.20 Aci_Gatt_Disc_Prim_serv_By_Uuid

Table 172. Aci_Gatt_Disc_Prim_serv_By_Uuid

Command name	Parameters	Return
Aci_Gatt_Disc_Prim_Serv_By_Uuid (0xFD13)	Connection handle UUID type UUID	Status

Description:

This command will start the procedure to discover the primary services of the specified UUID on the server.

Table 173. Aci_Gatt_Disc_Prim_serv_By_Uuid command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
UUID type	1 byte	0x01: 16-bit UUID (default) 0x02: 128-bit UUID
UUID	2 bytes or 16 bytes	UUID

Table 174. Aci_Gatt_Disc_Prim_serv_By_Uuid return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. The responses of the procedure are given through the Evt_Blue_Att_Find_By_Type_Value_Resp event. The end of the procedure is indicated by a Evt_Blue_Procedure_Complete event.

4.5.21 Aci_Gatt_Find_Included_Services

Table 175. Aci_Gatt_Find_Included_Services

Command name	Parameters	Return
Aci_Gatt_Find_Included_Services (0xFD14)	Connection handle Start handle End handle	Status

Description:

Start the procedure to find all included services.

Table 176. Aci_Gatt_Find_Included_Services command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
Start handle	2 bytes	Start handle of the service
End handle	2 bytes	End handle of the service

Table 177. Aci_Gatt_Find_Included_Services return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. The responses of the procedure are given through the Evt_Blue_Att_Read_By_Type_Resp event. The end of the procedure is indicated by a Evt_Blue_Procedure_Complete event.

4.5.22 Aci_Gatt_Disc_All_Charac_Of_Serv

Table 178. Aci_Gatt_Disc_All_Charac_Of_Serv

Command name	Parameters	Return
Aci_Gatt_Disc_All_Charac_Of_Serv (0xFD15)	Connection handle Start Attr handle End Attr handle	Status

Description:

Start the procedure to discover all the characteristics of a given service.

Table 179. Aci_Gatt_Disc_All_Charac_Of_Serv command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
Start Attr handle	2 bytes	Start attribute handle of the service
End Attr handle	2 bytes	End attribute handle of the service

Table 180. Aci_Gatt_Disc_All_Charac_Of_Serv return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packets are given through Evt_Blue_Att_Read_By_Type_Resp event.

4.5.23 Aci_Gatt_Disc_Charac_By_UUID

Table 181. Aci_Gatt_Disc_Charac_By_UUID

Command name	Parameters	Return
Aci_Gatt_Disc_Charac_By_UUID (0xFD16)	Connection handle Start Attr handle End Attr handle UUID type UUID	Status

Description:

Start the procedure to discover all the characteristics specified by the UUID.

Table 182. Aci_Gatt_Disc_Charac_By_UUID command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
Start Attr handle	2 bytes	Start attribute handle of the service
End Attr handle	2 bytes	End attribute handle of the service
UUID type	1 byte	0x01: 16-bit UUID (default) 0x02: 128-bit UUID
UUID	2 bytes or 16 bytes	UUID

Table 183. Aci_Gatt_Disc_Charac_By_UUID return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packets are given through Evt_Blue_Gatt_Disc_Read_Charac_By_UUID_Resp event.

4.5.24 Aci_Gatt_Disc_All_Charac_Descriptors

Table 184. Aci_Gatt_Disc_All_Charac_Descriptors

Command name	Parameters	Return
Aci_Gatt_Disc_All_Charac_Descriptors (0xFD17)	Connection handle charValHandle EndHandle	Status

Description:

Start the procedure to discover all characteristic descriptors on the server.

Table 185. Aci_Gatt_Disc_All_Charac_Descriptors command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
charValHandle	2 bytes	Starting handle of the characteristic
EndHandle	2 bytes	End handle of the characteristic

Table 186. Aci_Gatt_Disc_All_Charac_Descriptors return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packets are given through Evt_Blue_Att_Find_Info_Resp event.

4.5.25 Aci_Gatt_Read_Charac_Val

Table 187. Aci_Gatt_Read_Charac_Val

Command name	Parameters	Return
Aci_Gatt_Read_Charac_Val (0xFD18)	Connection handle attrHandle	Status

Description:

Start the procedure to read the attribute value.

Table 188. Aci_Gatt_Read_Charac_Val command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
attrHandle	2 bytes	Handle of the characteristic to be read

Table 189. Aci_Gatt_Read_Charac_Val return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packet is given through Evt_Blue_Gatt_Disc_Read_Charac_By_UUID_Resp event.

4.5.26 Aci_Gatt_Read_Charac_Using_UUID

Table 190. Aci_Gatt_Read_Charac_Using_UUID

Command name	Parameters	Return
Aci_Gatt_Read_Charac_Using_UUID (0xFD19)	Connection handle Start handle End handle UUID type UUID	Status

Description:

Start the procedure to read all the characteristics specified by the UUID.

Table 191. Aci_Gatt_Read_Charac_Using_UUID command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
Start handle	2 bytes	Starting handle of the range to be searched
End handle	2 bytes	End handle of the range to be searched
UUID type	1 byte	0x01: 16-bit UUID (default) 0x02: 128-bit UUID
UUID	2 bytes or 16 bytes	UUID

Table 192. Aci_Gatt_Read_Charac_Using_UUID return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packets are given through Evt_Blue_Gatt_Disc_Read_Charac_By_UUID_Resp event.

4.5.27 Aci_Gatt_Read_Long_Charac_Val

Table 193. Aci_Gatt_Read_Long_Charac_Val

Command name	Parameters	Return
Aci_Gatt_Read_Long_Charac_Val (0xFD1A)	Connection handle attrHandle valOffset	Status

Description:

Start the procedure to read a long characteristic value.

Table 194. Aci_Gatt_Read_Long_Charac_Val command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
attrHandle	1 byte	Handle of the characteristic to be read
valOffset	2 bytes	Offset from which the value needs to be read

Table 195. Aci_Gatt_Read_Long_Charac_Val command return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packets are given through Evt_Blue_Att_Read_Blob_Resp event.

4.5.28 Aci_Gatt_Read_Multiple_Charac_Values

Table 196. Aci_Gatt_Read_Multiple_Charac_Values

Command name	Parameters	Return
Aci_Gatt_Read_Multiple_Charac_Values (0xFD1B)	Connection handle numHandles setOfHandles	Status

Description:

Post the Read Multiple request.

Table 197. Aci_Gatt_Read_Multiple_Charac_Values command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
numHandles	1 byte	The number of handles for which the value has to be read
setOfHandles	numHandle * 2 bytes	The handles for which the attribute value has to be read

Table 198. Aci_Gatt_Read_Multiple_Charac_Values return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packets are given through Evt_Blue_Att_Read_Multiple_Resp event.

4.5.29 Aci_Gatt_Write_Charac_Val

Table 199. Aci_Gatt_Write_Charac_Val

Command name	Parameters	Return
Aci_Gatt_Write_Charac_Val (0xFD1C)	Connection handle attrHandle ValLen AttrVal	Status

Description:

Start the procedure to write a characteristic value.

Table 200. Aci_Gatt_Write_Charac_Val command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
attrHandle	2 bytes	Handle of the characteristic to be written
ValLen	1 byte	Length of the value to be written
AttrVal	0-N bytes	Value to be written

Table 201. Aci_Gatt_Write_Charac_Val return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated.

4.5.30 Aci_Gatt_Write_Long_Charac_Val

Table 202. Aci_Gatt_Write_Long_Charac_Val

Command name	Parameters	Return
Aci_Gatt_Write_Long_Charac_Val (0xFD1D)	Connection handle AttrHandle ValOffset ValLen AttrVal	Status

Description:

Start the procedure to write a long characteristic value.

Table 203. Aci_Gatt_Write_Long_Charac_Val command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
AttrHandle	2 bytes	Handle of the attribute to be written
ValOffset	2 bytes	Offset at which the attribute has to be written
ValLen	1 byte	Length of the value to be written
AttrVal	0-N bytes	Value to be written

Table 204. Aci_Gatt_Write_Long_Charac_Val return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packets are given through Evt_Blue_Att_Prepare_Write_Resp and Evt_Blue_Att_Exec_Write_Resp events.

4.5.31 Aci_Gatt_Write_Charac_Reliable

Table 205. Aci_Gatt_Write_Charac_Reliable

Command name	Parameters	Return
Aci_Gatt_Write_Charac_Reliable (0xFD1E)	Connection handle AttrHandle ValOffset ValLen AttrVal	Status

Description:

Start the procedure to write a characteristic reliably.

Table 206. Aci_Gatt_Write_Charac_Reliable command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
AttrHandle	2 bytes	Handle of the attribute to be written
ValOffset	2 bytes	Offset at which the attribute has to be written
ValLen	1 byte	Length of the value to be written
AttrVal	0-N bytes	Value to be written

Table 207. Aci_Gatt_Write_Charac_Reliable return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packets are given through Evt_Blue_Att_Prepare_Write_Resp and Evt_Blue_Att_Exec_Write_Resp events.

4.5.32 Aci_Gatt_Write_Long_Charac_Descriptor

Table 208. Aci_Gatt_Write_Long_Charac_Descriptor

Command name	Parameters	Return
Aci_Gatt_Write_Long_Charac_Descriptor (0xFD1F)	Connection handle attrHandle ValOffset ValLen AttrVal	Status

Description:

Start the procedure to write a long characteristic descriptor.

Table 209. Aci_Gatt_Write_Long_Charac_Descriptor command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
attrHandle	2 bytes	Handle of the attribute to be written
ValOffset	2 bytes	Offset at which the attribute has to be written
ValLen	1 byte	Length of the value to be written
AttrVal	0-N bytes	Value to be written

Table 210. Aci_Gatt_Write_Long_Charac_Descriptor return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packets are given through and Evt_Blue_Att_Prepare_Write_Resp and Evt_Blue_Att_Exec_Write_Resp event.

4.5.33 Aci_Gatt_Read_Long_Charac_Descriptor

Table 211. Aci_Gatt_Read_Long_Charac_Descriptor

Command name	Parameters	Return
Aci_Gatt_Read_Long_Charac_Descriptor (0xFD20)	Connection handle attrHandle valOffset	Status

Description:

Start the procedure to read a long characteristic value.

Table 212. Aci_Gatt_Read_Long_Charac_Descriptor command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
attrHandle	2 bytes	Handle of the characteristic descriptor
ValOffset	2 bytes	Offset from which the value needs to be read

Table 213. Aci_Gatt_Read_Long_Charac_Descriptor return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packets are given through Evt_Blue_Att_Read_Blob_Resp event.

4.5.34 Aci_Gatt_Write_Charac_Descriptor

Table 214. Aci_Gatt_Write_Charac_Descriptor

Command name	Parameters	Return
Aci_Gatt_Write_Charac_Descriptor (0xFD21)	Connection handle AttrHandle ValLen AttrVal	Status

Description:

Start the procedure to write a characteristic value.

Table 215. Aci_Gatt_Write_Charac_Descriptor command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
AttrHandle	2 bytes	Handle of the attribute to be written
ValLen	1 byte	Length of the value to be written
AttrVal	0-N bytes	Value to be written

Table 216. Aci_Gatt_Write_Charac_Descriptor return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated.

4.5.35 Aci_Gatt_Read_Charac_Descriptor

Table 217. Aci_Gatt_Read_Charac_Descriptor

Command name	Parameters	Return
Aci_Gatt_Read_Charac_Descriptor (0xFD22)	Connection handle AttrHandle	Status

Description:

Start the procedure to read the descriptor specified.

Table 218. Aci_Gatt_Read_Charac_Descriptor command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
AttrHandle	2 bytes	Handle of the descriptor to be read

Table 219. Aci_Gatt_Read_Charac_Descriptor return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packet is given through Evt_Blue_Att_Read_Resp event.

4.5.36 Aci_Gatt_Write_Without_Response

Table 220. Aci_Gatt_Write_Without_Response

Command name	Parameters	Return
Aci_Gatt_Write_Without_Response (0xFD23)	Connection handle AttrHandle ValLen AttrVal	Status

Description:

Start the procedure to write a characteristic value without waiting for any response from the server.

Table 221. Aci_Gatt_Write_Without_Response command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
AttrHandle	2 bytes	Handle of the attribute to be written
ValLen	1 byte	Length of the value to be written
AttrVal	0-N bytes	Value to be written

Table 222. Aci_Gatt_Write_Without_Response return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command complete event is generated when this command is processed.

4.5.37 Aci_Gatt_Signed_Write_Without_Response

Table 223. Aci_Gatt_Signed_Write_Without_Response

Command name	Parameters	Return
Aci_Gatt_Signed_Write_Without_Response (0xFD24)	Connection handle AttrHandle ValLen AttrVal	Status

Description:

Start the procedure to write a characteristic value with an authentication signature without waiting for any response from the server. It cannot be used when the link is encrypted.

Table 224. Aci_Gatt_Signed_Write_Without_Response command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
AttrHandle	2 bytes	Handle of the attribute to be written
ValLen	1 byte	Length of the value to be written
AttrVal	0-N bytes	Value to be written

Table 225. Aci_Gatt_Signed_Write_Without_Response return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command complete event is generated when this command is processed.

4.5.38 Aci_Gatt_Confirm_Indication

Table 226. Aci_Gatt_Confirm_Indication

Command name	Parameters	Return
Aci_Gatt_Confirm_Indication (0xFD25)	Connection handle	Status

Description:

Allow application to confirm indication. This command has to be sent when the application receives the event Evt_Blue_Gatt_Indication.

Table 227. Aci_Gatt_Confirm_Indication command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given

Table 228. Aci_Gatt_Confirm_Indication return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x46: Not allowed - If the exchange has already taken place - Channel not open

Event(s) generated:

A command complete event is generated when this command is processed.

4.5.39 Aci_Gatt_Write_Resp

Table 229. Aci_Gatt_Write_Resp

Command name	Parameters	Return
Aci_Gatt_Write_Resp (0xFD26)	Connection handle Attribute handle Status Error Code Att_Val_Len Att_Val	Status

Description:

This command has to be sent by the application when it receives the Evt_Blue_Gatt_Write_Permit_req. If the write can be allowed, then the status and error code has to be set to 0. If the write cannot be allowed, then the status has to be set to 1 and the error code has to be set to the error code that has to be passed to the client.

Table 230. Aci_Gatt_Write_Resp command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given
Attribute handle	2 bytes	Handle of the attribute that was passed in the event Evt_Blue_Gatt_Write_Permit_req
Status	1 byte	0x00: The value can be written to the attribute specified by handle 0x01: The value should not be written to the attribute specified by the handle
Error Code	1 byte	The error code that has to be passed to the client in case the write has to be rejected
Att_Val_Len	1 byte	Length of the value to be written as passed in the event Evt_Blue_Gatt_Write_Permit_req
Att_Val	0-N bytes	Value as passed in the event Evt_Blue_Gatt_Write_Permit_req.

Table 231. Aci_Gatt_Write_Resp return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x46: Not allowed The command is not allowed in the current state of stack

Event(s) generated:

A command complete event is generated when this command is processed.

4.5.40 Aci_Gatt_Allow_Read

Table 232. Aci_Gatt_Allow_Read

Command name	Parameters	Return
Aci_Gatt_Allow_Read (0xFD27)	Connection handle	Status

Description:

The application has to send this command when it receives the Evt_Blue_Gatt_Read_Permit_Req or Evt_Blue_Gatt_Read_Multi_Permit_Req. This command indicates to the stack that the response can be sent to the client. So if the application wishes to update any of the attributes before they are read by the client, it has to update the characteristic values using the Aci_Gatt_Update_Charac_Value and then give this command. The application should perform the required operations within 30 seconds. Otherwise the GATT procedure will be timeout.

Table 233. Aci_Gatt_Allow_Read command parameters

Parameter	Size	Description
Connection handle	2 bytes	Connection handle for which the command is given

Table 234. Aci_Gatt_Allow_Read return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x46: Not allowed The command is not expected in the current state of the stack

Event(s) generated:

A command complete event is generated when this command is processed.

4.5.41 Aci_Gatt_Set_Security_Permission

Table 235. Aci_Gatt_Set_Security_Permission

Command name	Parameters	Return
Aci_Gatt_Set_Security_Permission (0xFD28)	Service handle Attr handle Security permission	Status

Description:

This command sets the security permission for the attribute handle specified. Currently the setting of security permission is allowed only for client configuration descriptor.

Table 236. Aci_Gatt_Set_Security_Permission command parameters

Parameter	Size	Description
Service handle	2 bytes	Handle of the service which contains the attribute whose security permission has to be modified.
Attr handle	2 bytes	Handle of the attribute whose security permission has to be modified.
Security permission	1 byte	0x00: ATTR_PERMISSION_NONE 0x01: ATTR_PERMISSION_AUTHEN_READ 0x02: ATTR_PERMISSION_AUTHOR_READ 0x04: ATTR_PERMISSION_ENCRY_READ 0x08: ATTR_PERMISSION_AUTHEN_WRITE 0x10: ATTR_PERMISSION_AUTHOR_WRITE 0x20: ATTR_PERMISSION_ENCRY_WRITE

Table 237. Aci_Gatt_Set_Security_Permission return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x12: Invalid parameters 0x41: Failed Could not find the handle specified or the handle is not of a client configuration descriptor

Event(s) generated:

A command complete event is generated when this command is processed.

4.5.42 Aci_Gatt_Set_Descriptor_Value

Table 238. Aci_Gatt_Set_Descriptor_Value

Command name	Parameters	Return
Aci_Gatt_Set_Descriptor_Value (0xFD29)	Service_Handle Char_Handle Desc_Handle val_offset desc_val_len desc_val	Status

Description:

This command sets the value of the descriptor specified by Desc_handle.

Table 239. Aci_Gatt_Set_Descriptor_Value command parameters

Parameter	Size	Description
Service handle	2 bytes	Handle of the service which contains the descriptor.
Char handle	2 bytes	Handle of the characteristic which contains the descriptor.
Desc handle	2 bytes	Handle of the descriptor whose value has to be set.
val_offset	2 bytes	Offset from which the descriptor value has to be updated.
desc_val_len	2 bytes	Length of the descriptor value
desc_val	0-N bytes	Descriptor value

Table 240. Aci_Gatt_Set_Descriptor_Value return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x47: Error 0x60: Invalid handle 0x61: Invalid parameter

Event(s) generated:

A command complete event is generated when this command is processed.

4.5.43 Aci_Gatt_Read_Handle_Value

Table 241. Aci_Gatt_Read_Handle_Value

Command name	Parameters	Return
Aci_Gatt_Read_Handle_Value (0xFD2A)	Attribute handle	Status value

Description:

Reads the value of the attribute handle specified.

Table 242. Aci_Gatt_Read_Handle_Value command parameters

Parameter	Size	Description
Attribute handle	2 bytes	Handle of the attribute to read

Table 243. Aci_Gatt_Read_Handle_Value return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x41: BLE_STATUS_FAILED 0x12: ERR_INVALID_HCI_CMD_PARAMS
Length	2 bytes	Length of the value
Value	0-N bytes	Value read. The length is as specified in the Length field.

Event(s) generated:

A command complete event is returned when this command is received. The event will contain the length and value of the attribute handle specified.

4.6 GATT VS events

Table 244. GATT VS events

Item	Event	EGID	EID	ECODE
1	Evt_Blue_Gatt_Attribute_modified	0x03	0x01	0x0C01
2	Evt_Blue_Gatt_Procedure_Timeout	0x03	0x02	0x0C02
3	Evt_Blue_Att_Exchange_MTU_Resp	0x03	0x03	0x0C03
4	Evt_Blue_Att_Find_Information_Resp	0x03	0x04	0x0C04
5	Evt_Blue_Att_Find_By_Type_Value_Resp	0x03	0x05	0x0C05
6	Evt_Blue_Att_Read_By_Type_Resp	0x03	0x06	0x0C06
7	Evt_Blue_Att_Read_Resp	0x03	0x07	0x0C07
8	Evt_Blue_Att_Read_Blob_Resp	0x03	0x08	0x0C08
9	Evt_Blue_Att_Read_Multiple_Resp	0x03	0x09	0x0C09
10	Evt_Blue_Att_Read_By_Group_Type_Resp	0x03	0x0A	0x0C0A
11	Evt_Blue_Att_Write_Resp	0x03	0x0B	0x0C0B
12	Evt_Blue_Att_Prepare_Write_Resp	0x03	0x0C	0x0C0C
13	Evt_Blue_Att_Exec_Write_Resp	0x03	0x0D	0x0C0D
14	Evt_Blue_Gatt_Indication	0x03	0x0E	0x0C0E
15	Evt_Blue_Gatt_notification	0x03	0x0F	0X0C0F
16	Evt_Blue_Gatt_Procedure_Complete	0x03	0x10	0X0C10
17	Evt_Blue_Gatt_Error_Response	0x03	0x11	0X0C11
18	Evt_Blue_Gatt_Disc_Read_Charac_By_UUID_Resp	0x03	0x12	0X0C12
19	Evt_Blue_Gatt_Write_Permit_req	0x03	0x13	0X0C13
20	Evt_Blue_Gatt_Read_Permit_Req	0x03	0x14	0X0C14
21	Evt_Blue_Gatt_Read_Multi_Permit_Req	0x03	0x15	0X0C15

4.6.1 Evt_Blue_Gatt_Attribute_modified

This event is generated to the application by the GATT server when a client modifies any attribute on the server.

Table 245. Evt_Blue_Gatt_Attribute_modified

Parameter	Size	Description
Event code	2 bytes	The event code of the vendor specific event
Connection handle	2 bytes	The connection handle which modified the attribute
Attribute handle	2 bytes	Handle of the attribute that was modified
data_len	1 byte	The length of the data to follow
data buffer	0-N bytes	The modified data

4.6.2 Evt_Blue_Gatt_Procedure_Timeout

This event is generated by the client/server to the application on a GATT timeout (30 seconds).

Table 246. Evt_Blue_Gatt_Procedure_Timeout

Parameter	Size	Description
Event code	2 bytes	The event code of the vendor specific event
Connection handle	2 bytes	The connection handle on which the gatt procedure has timed out

4.6.3 Evt_Blue_Gatt_Procedure_Complete

This event is generated when a GATT client procedure completes either with error or successfully.

Table 247. Evt_Blue_Gatt_Procedure_Complete

Parameter	Size	Description
Event code	2 bytes	The event code of the vendor specific event
Connection handle	2 bytes	The connection handle which the gatt procedure has completed
data_len	1 byte	Will always be 1 byte.
data buffer	0-N bytes	Indicates whether the procedure completed with error (BLE_STATUS_FAILED) or was successful (BLE_STATUS_SUCCESS).

4.6.4 Evt_Blue_Gatt_Disc_Read_Charac_By_UUID_Resp

This event can be generated during a "Discover Characteristics By UUID" procedure or a "Read using Characteristic UUID" procedure.

The attribute value will be a service declaration as defined in Bluetooth Core v4.0 spec (vol.3, Part G, ch. 3.3.1), when a "Discover Characteristics By UUID" has been started. It will be the value of the Characteristic if a* "Read using Characteristic UUID" has been performed.

Table 248. Evt_Blue_Gatt_Disc_Read_Charac_By_UUID_Resp

Parameter	Size	Description
Event_code	2 bytes	The event code of the vendor specific event
Connection_Handle	2 bytes	Connection handle where the procedure started.
Data_len	1 byte	Length of following data
Attribute_Handle	2 bytes	Handle of the discovered attribute
Attribute_Value	(Data_len - 2) bytes	Attribute value

4.6.5 Evt_Blue_Gatt_Write_Permit_req

This event is given to the application when a write request, write command or signed write command is received by the server from the client. This event will be given to the application only if the event bit for this event generation is set when the characteristic was added.

When this event is received, the application has to check whether the value being requested for write can be allowed to be written and respond with the command Aci_Gatt_Write_Resp. The details of the parameters of the command can be found. Based on the response from the application, the attribute value will be modified by the stack. If the write is rejected by the application, then the value of the attribute will not be modified. In case of a write REQ, an error response will be sent to the client, with the error code as specified by the application. In case of write/signed write commands, no response is sent to the client but the attribute is not modified.

Table 249. Evt_Blue_Gatt_Write_Permit_req

Parameter	Size	Description
Event code	2 bytes	The event code of the vendor specific event
Connection handle	2 bytes	Handle of the connection on which there was the request to write the attribute
Attribute handle	2 bytes	The handle of the attribute for which the write request has been made by the client
data_len	1 byte	The length of the data to follow
data buffer	0-N bytes	The data that the client has requested to write

4.6.6 Evt_Blue_Gatt_Read_Permit_Req

This event is given to the application when a read request or read blob request is received by the server from the client. This event will be given to the application only if the event bit for this event generation is set when the characteristic was added.

On receiving this event, the application can update the value of the handle if it desires and when done, it has to send the Aci_Gatt_Allow_Read command to indicate to the stack that it can send the response to the client.

Table 250. Evt_Blue_Gatt_Read_Permit_Req

Parameter	Size	Description
Event code	2 bytes	The event code of the vendor specific event
Connection handle	2 bytes	Handle of the connection on which there was the request to read the attribute
Attribute handle	2 bytes	The handle of the attribute that has been requested by the client to be read
Datalen	1 byte	Length of the data to follow
Data	0-N bytes	Contains the offset from which the read has been requested

4.6.7 Evt_Blue_Gatt_Read_Multi_Permit_Req

This event is given to the application when a read multiple request or read by type request is received by the server from the client. This event will be given to the application only if the event bit for this event generation is set when the characteristic was added.

On receiving this event, the application can update the values of the handles if it desires and when done, it has to send the Aci_Gatt_Allow_Read command to indicate to the stack that it can send the response to the client.

Table 251. Evt_Blue_Gatt_Read_Multi_Permit_Req

Parameter	Size	Description
Event code	2 bytes	The event code of the vendor specific event
Connection handle	2 bytes	Handle of the connection which requested to read the attribute
data_len	1 byte	The length of the data to follow
data buffer	0-N bytes	The handles of the attributes that have been requested by the client for a read

4.6.8 Other GATT VS events

Table 252. Other GATT VS events

Parameter	Size	Description
Event code	2 bytes	The event code of the vendor specific event
Connection handle	2 bytes	Connection handle on which the gatt procedure is running
data_len	1 byte	The length of the data to follow if any
data buffer	0-N bytes	Indicates whether the procedure completed with error (BLE_STATUS_FAILED) or was successful (BLE_STATUS_SUCCESS). This event will be generated when any one of the GATT client procedures are started by the application. The application has to associate the event with the procedure it had started.

4.7 HCI Vendor specific commands

4.7.1 HCI VS commands

Table 253. HCI VS commands

Item	Event	EGID	EID	ECODE
1	Hal_IO_Init	0x00	0x01	0xFC01
2	Hal_IO_Configure	0x00	0x02	0xFC02
3	Hal_IO_set_bit	0x00	0x03	0xFC03
4	Hal_IO_get_bit	0x00	0x04	0xFC04
5	Write_Config_data	0x00	0x0C	0xFC0C
6	Read_Config_data	0x00	0x0D	0xFC0D
7	Set_Tx_power_level	0x00	0x0F	0xFC0F
8	Device_Standby	0x00	0x13	0xFC13
9	Number_Of_Packet_Sent_in_DirectTX	0x00	0x14	0xFC14
10	Tone_Start	0x00	0x15	0xFC15
11	Tone_Stop	0x00	0x16	0xFC16
12	Updater_Start	0x00	0x20	0xFC20
13	Updater_Reboot	0x00	0x21	0xFC21
14	Updater_Version	0x00	0x22	0xFC22
15	Updater_Buffer_Size	0x00	0x23	0xFC23
16	Updater_Erase_Blue_Flag	0x00	0x24	0xFC24
17	Updater_Reset_Blue_Flag	0x00	0x25	0xFC25
18	Updater_Erase_Sector	0x00	0x26	0xFC26
19	Updater_Program_Data_Block	0x00	0x27	0xFC27
20	Updater_Read_Data_Block	0x00	0x28	0xFC28
21	Updater_Calculate_CRC	0x00	0x29	0xFC29
22	Updater_Read_HW_Version	0x00	0x2A	0xFC2A

1. For the updater commands, i.e. OpCode with 0xFC2x, please refer to the separate document BlueNRG Updater Specification.
2. The 4 HAL_IO_XXX commands (Item 1-4) are disabled in the current release.

4.7.2 Hal_IO_Init

Table 254. Hal_IO_Init

Command name	Parameters	Return
Hal_IO_Init (0xFC01)		Status

Description:

Initialize the GPIOs of the BlueNRG device.

Table 255. Hal_IO_Init return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x47: BLE_STATUS_ERROR

Event(s) generated:

The controller will generate a command complete event.

4.7.3 Hal_IO_Configure

Table 256. Hal_IO_Configure

Command name	Parameters	Return
Hal_IO_Configure (0xFC02)	OE	Status

Description:

Set the direction of the GPIOs.

Table 257. Hal_IO_Configure command parameters

Parameter	Size	Description
OE	1 byte	Each bit represent one IO, a value “1” is output and a “0” input

Table 258. Hal_IO_Configure return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x47: BLE_STATUS_ERROR

Event(s) generated:

The controller will generate a command complete event.

4.7.4 Hal_IO_set_bit

Table 259. Hal_IO_set_bit

Command name	Parameters	Return
Hal_IO_set_bit (0xFC03)	Index Value	Status

Description:

Set the value of a given IO.

Table 260. Hal_IO_set_bit command parameters

Parameter	Size	Description
Index	1 byte	The index of the pin to be manipulated.
Value	1 byte	A value equal to “0” reset the state of the pin; a different value set the pin to high.

Table 261. Hal_IO_set_bit return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x47: BLE_STATUS_ERROR

Event(s) generated:

The controller will generate a command complete event.

4.7.5 Hal_IO_get_bit

Table 262. Hal_IO_get_bit

Command name	Parameters	Return
Hal_IO_get_bit (0xFC04)	Index	Status Value

Description:

Request the value of a pin.

Table 263. Hal_IO_get_bit command parameters

Parameter	Size	Description
Index	1 byte	The index of the pin to be manipulated.

Table 264. Hal_IO_get_bit return parameters

Parameter	Size	Description
Index	1 byte	0x00: BLE_STATUS_SUCCESS 0x47: BLE_STATUS_ERROR
Value	1 byte	0x0: IO value is 0 0x1: IO value is 1

Event(s) generated:

The controller will generate a command complete event.

4.7.6 Write_Config_Data

Table 265. Write_Config_Data

Command name	Parameters	Return
Write_Config_Data (0xFC0C)	Offset Length Value	Status

Description:

This command writes a value to a low level configure data structure. It is useful to setup directly some low level parameters for the system in the runtime.

Table 266. Write_Config_Data command parameters

Parameter	Size	Description
Offset	1 byte	Offset in the data structure. The starting member in the data structure will have an offset 0.
Length	1 byte	Length of data to be written
Value	0-N bytes	Data to be written

Table 267. Write_Config_Data members

Data members	Size	Offset	Description
Public address	6 bytes	0x00	Bluetooth public address
DIV	2 bytes	0x06	DIV used to derive CSRK
ER	16 bytes	0x08	Encryption root key used to derive LTK and CSRK
IR	16 bytes	0x18	Identity root key used to derive LTK and CSRK

Table 267. Write_Config_Data members (continued)

Data members	Size	Offset	Description
LLWithoutHost	1 byte	0x2C	Switch on/off Link Layer only mode
Role	1 byte	0x2D	Select the BlueNRG roles and mode configuration. 1. Slave and master (only one connection), only 6 KB of RAM retention. 2. Slave and master (only one connection), 12 KB of RAM retention 3. Master (up to 8 slaves), 12 KB of RAM retention

Table 268. Write_Config_Data return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: ERR_INVALID_HCI_CMD_PARAMS

Event(s) generated:

The controller will generate a command complete event.

4.7.7 Read_Config_Data

Table 269. Read_Config_Data

Command name	Parameters	Return
Read_Config_Data (0xFC0D)	Offset	Status Value

Description:

This command requests the value in the low level configure data structure. For more information see the command Write_Config_Data. The number of bytes of returned Value changes for different Offset.

Table 270. Read_Config_Data command parameters

Parameter	Size	Description
Offset	1 byte	Offset in the data structure. The starting member in the data structure will have an offset 0.

Table 271. Read_Config_Data return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: ERR_INVALID_HCI_CMD_PARAMS
Value	0-N bytes	Value read at the offset specified

Event(s) generated:

The controller will generate a command complete event.

4.7.8 Set_Tx_power_level

Table 272. Set_Tx_power_level

Command name	Parameters	Return
Set_Tx_power_level (0xFC0F)	EN_HIGH_POWER PA_LEVEL	Status

Description:

This command sets the TX power level of the BlueNRG. By controlling the EN_HIGH_POWER and the PA_LEVEL, the combination of the 2 determines the output power level (dBm). See the table below.

When the system starts up or reboots, the default TX power level will be used, which is the maximum value of 8 dBm. Once this command is given, the output power will be changed instantly, regardless if there is Bluetooth communication going on or not. For example, for debugging purpose, the BlueNRG can be set to advertise all the time. And use this command to observe the signal strength changing.

The system will keep the last received TX power level from the command, i.e. the 2nd command overwrites the previous TX power level. The new TX power level remains until another Set TX Power command, or the system reboots.

Table 273. Set_Tx_power_level command parameters

Parameter	Size	Description
EN_HIGH_POWER	1 byte	Can be only 0 or 1. Set high power bit on or off.
PA_LEVEL	1 byte	Can be from 0 to 7. Set the PA level value.

Table 274. Tx_power_level command parameters combination

EN_HIGH_POWER	PA_LEVEL	TX Power Level (dBm)
0	0	-18
0	1	-14.7
0	2	-11.4
0	3	-8.1
0	4	-4.9
0	5	-1.6
0	6	1.7
0	7	5.0
1	0	-15
1	1	-11.7

Table 274. Tx_power_level command parameters combination (continued)

EN_HIGH_POWER	PA_LEVEL	TX Power Level (dBm)
1	2	-8.4
1	3	-5.1
1	4	-2.1
1	5	1.4
1	6	4.7
1	7	8.0 (default)

Table 275. Set_Tx_power_level return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: ERR_INVALID_HCI_CMD_PARAMS

Event(s) generated:

The controller will generate a command complete event.

4.7.9 Device_Standby

Table 276. Device_Standby

Command name	Parameters	Return
Device_Standby(0xFC13)		Status

Description:

Normally the BlueNRG will automatically enter sleep mode to save power. This Device_Standby command further put the device into the Standby mode instead of the sleep mode. The difference is that, in sleep mode, the device can still wake up itself with the internal timer. But in standby mode, this timer is also disabled. So the only possibility to wake up the device is by the external signals, e.g. a HCI command sent via SPI bus.

Based on the measurement, the current consumption under sleep mode is ~2 uA. And this value is ~1.5 uA in standby mode.

The command is only accepted when there is no other Bluetooth activity. Otherwise an error code “command disallowed” will return.

Table 277. Device_Standby return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x0C: ERR_COMMAND_DISALLOWED This command is not allowed when Bluetooth activity is ongoing

Event(s) generated:

The controller will generate a command complete event.

4.7.10 Number_Of_Packet_Sent_in_DirectTX

Table 278. Number_Of_Packet_Sent_in_DirectTX

Command name	Parameters	Return
Number_Of_Packet_Sent_in_DirectTX (0xFC14)		Status Packet Counter

Description:

During the Direct Test mode, in the TX tests, the number of packets sent in the test is not returned when executing the Direct Test End command. This command implements this feature.

If the Direct TX test is started, a 32-bit counter will be used to count how many packets have been transmitted. After the Direct Test End, this command can be used to check how many packets were sent during the Direct TX test.

The counter starts from 0 and counts upwards. As would be the case if 32-bits are all used, the counter wraps back and starts from 0 again. The counter is not cleared until the next Direct TX test starts.

Table 279. Number_Of_Packet_Sent_in_DirectTX return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS
Packet counter	4 bytes	Number of packets sent during the last Direct TX test

Event(s) generated:

The controller will generate a command complete event.

4.7.11 Tone_Start

Table 280. Tone_Start

Command name	Parameters	Return
Tone_Start (0xFC15)	Channel ID	Status

Description:

This command starts a carrier frequency, i.e. a tone, on a specific channel. The frequency sine wave at the specific channel may be used for debugging purpose only. The channel ID is a parameter from 0x00 to 0x27 for the 40 BLE channels, e.g. 0x00 for 2.402 GHz, 0x01 for 2.404 GHz etc.

This command should not be used when normal Bluetooth activities are ongoing.

The tone should be stopped by Tone_Stop command.

Table 281. Tone_Start command parameters

Parameter	Size	Description
Channel ID	1 byte	BLE Channel ID, from 0x00 to 0x27 meaning (2.402 + 2*0xXX) GHz

Table 282. Tone_Start return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS

Event(s) generated:

The controller will generate a command complete event.

4.7.12 Tone_Stop

Table 283. Tone_Stop

Command name	Parameters	Return
Tone_Stop (0xFC16)		Status

Description:

This command is used to stop the previously started Tone_Start command.

Table 284. Tone_Stop return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS

Event(s) generated:

The controller will generate a command complete event.

5 SPI interface

The BlueNRG device provides an SPI interface, which can be used to connect an external device. The external device normally is an application processor. It sends ACI commands to control the BlueNRG device. And the BlueNRG generates ACI events to report back to the external application processor. The ACI commands and events are transmitted through the SPI bus.

This chapter describes the BlueNRG hardware SPI interface. Also it describes the communication protocol over the SPI bus between the BlueNRG and the external device.

5.1 Hardware SPI interface

The BlueNRG SPI interface is Motorola protocol compliant, which has 5 wires: CLK (clock), nCS (chip select), MOSI (master output slave input), MISO (master input slave output), and DataRdy_IRQ (transmit interrupt request).

The BlueNRG always behaves as the slave device on the SPI bus. The external device must be the SPI master.

The table below shows the 5 SPI pins on the BlueNRG device, including the pin number of the silicon chip and the pin direction from the BlueNRG's side.

Table 285. SPI pins

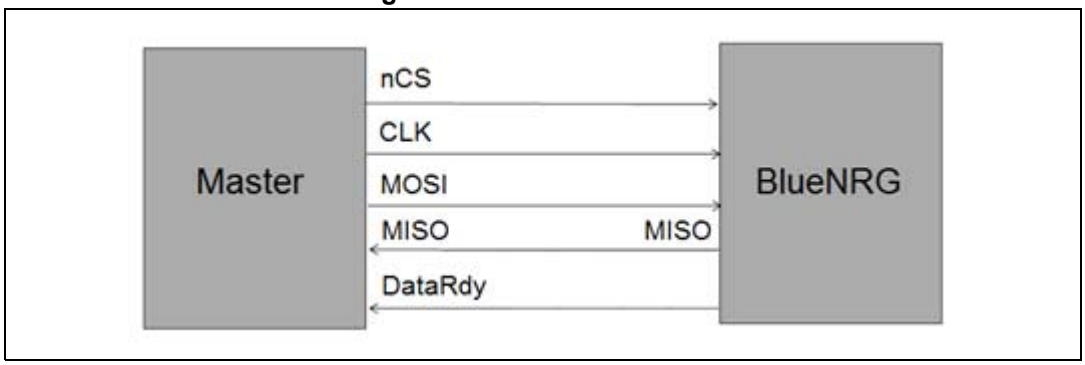
Pin name	Pin n°	Direction
SPI_CLK	2	Input
SPI_NCS	31	Input
SPI_MOSI	1	Input
SPI_MISO	32	Output
SPI_IRQ	3	Output

Here is the summary of the SPI parameters:

1. The CS line is active low.
2. The clock signal can go up to 8 MHz, or can go as low as 100 kHz.
3. The clock is inactive low, i.e. the clock polarity CPOL = 0.
4. The data is valid on clock leading edge, i.e. the clock phase CPHA = 0.
5. The data transfer is always 8-bit byte based.
6. For each byte, the Most Significant Bit is sent first.

Figure 9 shows how to connect BlueNRG SPI bus to an external SPI master.

Figure 9. SPI wire connection



The SPI master, i.e. an application processor, always controls when to start an SPI transaction. This can happen at any time. It is started by putting low the nCS line to activate the BlueNRG device. Then the master should provide the SPI clock. Together with the clock, the master should output the data on the SPI bus through the MOSI line, either dummy data or valid data. At the same time, the slave returns data from the MISO line, because SPI is in principle a shift register.

When the SPI master wants to send data to the BlueNRG, i.e. an SPI write, it sets nCS to low, then sends the SPI clock and dummy data to read an SPI header from the BlueNRG. The SPI header format will be described in the following section. If the SPI header indicates that the BlueNRG has enough space in the buffer, the SPI master can then send real data over the SPI bus. The SPI master should not overwrite the BlueNRG SPI buffer, otherwise the communication will fail.

When the BlueNRG wants to send data to the SPI master, i.e. an SPI read, it must use the IRQ pin, because the SPI slave cannot control the nCS line. The BlueNRG will set the IRQ pin to high, which notifies the SPI master that there is data to be read. Then the SPI master should start an SPI transaction, reading the SPI header, and from where it will know how many bytes it should read back from the BlueNRG. The SPI master should not read bytes other than those indicated as the SPI header, otherwise the communication will fail.

The DataRdy (IRQ) pin of the BlueNRG should work as follows: (1) it must be high when BlueNRG has data to send to the SPI master; (2) It must be high impedance when BlueNRG has no data to send to the SPI master. On the SPI master side, this pin must be configured as an input pin. The SPI master may poll this pin, or use it as an interrupt source, to detect when an SPI read is required. Also there should be a pull down resistor, either on the SPI master IRQ pin or externally connected. This pull down resistor makes sure the DataRdy value goes back to 0, so it does not confuse the SPI master to misread data. Note that the reason that IRQ pin must be high-Z when there is no data to send is because we also use this pin as input when BlueNRG boots up. The IRQ pin value will determine if the updater mode is required or not. After the BlueNRG goes to normal working state, the IRQ pin is configured as output for SPI interface.

Note also that the MISO also goes to high-Z when BlueNRG does not send data. This allows multi-SPI slaves on the same bus. This other SPI slave may access the bus when BlueNRG does not.

5.2 SPI communication protocol

To communicate with the BlueNRG, the data on the SPI bus must be formatted as described in this section.

An SPI transaction is defined from the time instant when the master puts the nCS line to low, until the time instant when the nCS line is back to high.

Each SPI transaction should contain 1 data frame. Each data frame should contain at least 5 bytes of header, and may have 0-N bytes of data.

5.2.1 Header

Figure 10. SPI header format

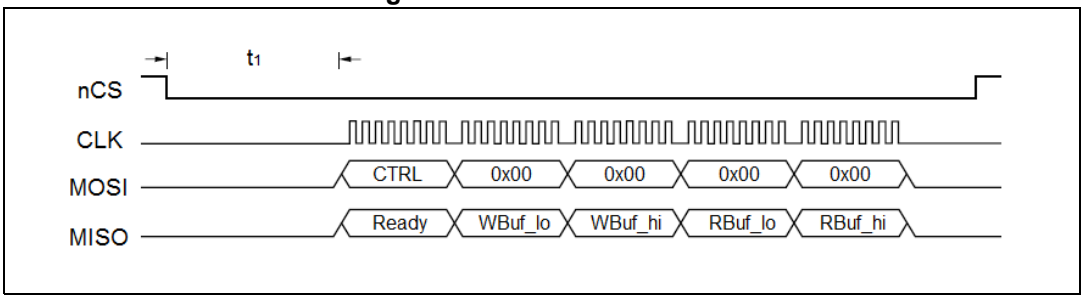


Figure 10 above shows an SPI frame, with minimum 5 bytes SPI header, and 0 byte SPI data.

The header of the master is on the MOSI line, which is 1 control byte and 4 bytes 0x00. The control byte can have only the value of 0x0A (SPI write) or 0x0B (SPI read).

The BlueNRG returns the slave header on the MISO line at the same time. The 1st byte is the SPI READY indication. It must be 0x02 to indicate the slave SPI interface is ready. If it is any value other than 0x02, the master must ignore the following 4 bytes and abort the SPI transaction. If the BlueNRG SPI is ready (Ready byte = 0x02), the following 4 bytes give 2 buffer sizes for write and read, 2 bytes for each buffer. Of the 2-byte buffer size, the lower byte comes first, then the higher byte. The write buffer size means how many bytes the master can write to the BlueNRG. The read buffer size means how many bytes in the BlueNRG are waiting for the master to read.

For example, if the slave header is 0x0200010500, the write buffer size is 0x0100 and the read buffer size is 0x0005. So the master cannot write more than 256 bytes to the BlueNRG, and it should read back 5 bytes from the BlueNRG as soon as possible.

The SPI master may poll the BlueNRG by sending dummy SPI headers, just to check what value is in the BlueNRG SPI header. For example, the SPI master can send an SPI frame with only 5 bytes header: the CTRL byte can be either write or read, then following 4 bytes 0x00 dummy bytes. After the header, the SPI master stops the frame, without sending any real data. So the write/read action will not happen in practice.

5.2.2 Write data to BlueNRG

To write data to the BlueNRG, the SPI master must start an SPI frame to check the BlueNRG SPI header, and the following 3 conditions must be met:

1. The master sends the CTRL byte as 0x0A.
2. The slave replies with SPI READY byte as 0x02.
3. The write buffer size is bigger than 0.

If so, the master can put the data bytes following the 5 header bytes on the MOSI line in the same transaction.

If the master is fast enough, it may start sending data bytes immediately in the same SPI transaction where the 3 conditions are just seen satisfied. Or the master may choose to stop the SPI transaction for now, go back to prepare the data, and send them in a later SPI transaction.

The SPI master is allowed to write less bytes than the write buffer size, but it is forbidden to write more. In case an ACI command packet is bigger than 128 bytes. The master has to send it in 2 SPI write transactions. The master firstly sends 128 bytes, then waiting for the write buffer to be available again, and then write the remaining bytes.

The BlueNRG will count how many bytes are actually received on the MOSI line. So the master does not need to indicate how many bytes it intends to write.

The write buffer size value is the data bytes of the payload. It does not include the 5 header bytes.

When the master is sending real data bytes on the MOSI line, the BlueNRG will return the same amount of bytes from its own shift register. The master should ignore these bytes.

5.2.3 Read data from BlueNRG

The BlueNRG uses the IRQ pin to notify the SPI master when it has data to be read. When the SPI master sees the IRQ pin become high, it should read data from BlueNRG.

The protocol is similar to SPI write. The SPI master starts a transaction to check the SPI header. The master puts 0xB in the CTRL byte, and checks again if the slave is ready (0x02), and the read buffer size is non-zero. Then the master sends certain amount of dummy bytes (e.g. 0x00 or 0xFF) on the MOSI line, while reading back the slave data from the MISO line. The reading process can happen in 1 SPI transaction or in multiple SPI transactions.

The SPI master should read from BlueNRG as long as the IRQ pin is high.

5.2.4 SPI operation with BlueNRG sleep mode

The BlueNRG device is designed with a deep sleep mode, in which it turns off most of the internal functions to save power. The BlueNRG goes into sleep mode automatically when there is no Bluetooth activity ongoing. So it is very likely that the BlueNRG is sleeping when the master starts an SPI transaction.

The BlueNRG wakes up as soon as the nCS line is put to low. But then it needs to initialize and prepare the SPI buffers. This will take some time. So the following sequence is likely to occur on the SPI bus:

1. The master puts the nCS line to low and the BlueNRG wakes up. BlueNRG is expecting a command within 2 ms before going to sleep.
2. The master sends the header, but the BlueNRG replies with the READY byte as 0x00 or 0xFF. This means the BlueNRG SPI interface is still not initialized.
3. The master keeps polling the BlueNRG, (releasing and asserting CS again). In order to read again the header, the SPI master has to release the CS line and assert it within 2ms from the previous CS assertion. After a while, the READY byte becomes 0x02 but both the buffer sizes are 0x00. This means the BlueNRG SPI is initialized, but the buffer is not yet prepared.
4. As soon as the buffer size becomes non-zero, the master can write data to the BlueNRG.

With the measurement, the master should allow about 0.5 ms for the BlueNRG to be completely ready for the SPI communication. Note that 0.5 ms is when non-retention RAM of BlueNRG must be initialized upon wakeup (i.e. device in mode1 - slave only mode). The value can be 0.3 ms if non-retention RAM initialization is not required, which makes the SPI ready faster.

This waiting operation is only needed when the SPI bus is idle for a while. If the master is continuously writing, from the 2nd write transaction there is no need to wait, because the BlueNRG will not sleep if there is still SPI data to be processed. Also, there is no need to wait when reading from the BlueNRG, because it cannot sleep if it still holds data for the master.

6 Revision history

Table 286. Document revision history

Date	Revision	Changes
30-May-2014	1	Initial release.
17-Nov-2014	2	Updated: Title, *Table 14*, *Table 15*, *Table 16*, *Table 17*, *Table 30*, *Table 54*, *Table 242*, minor changes in *Chapter 4: Vendor specific commands* and *Section 5.2.4* Added: *Section 4.3.35: Aci_Gap_Get_Bonded_Devices* and replaced technical content in *Table 113* Minor text changes.
01-Jul-2015	3	Updated: *Table 16*, *Table 38*, *Table 39*, *Table 85*, *Table 86*, *Table 88* and *Table 113*.

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2015 STMicroelectronics – All rights reserved